Connectivity Preservation in Multi-Agent Systems using Model Predictive Control

Ahmed El-Hamamsy

A Thesis
in
The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science (Electrical and Computer Engineering)
at Concordia University
Montreal, Quebec, Canada

December 2019

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Ahmed El-Hamamsy

Entitled: Connectivity Preservation in Multi-Agent Systems Using Model Predictive Control

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. H. Rivaz

_____ External Examiner
Dr. F. Aghili (MIAE)

_____ Internal Examiner
Dr. H. Rivaz

_____ Supervisor
Dr. A.G. Aghdam

Approved by: _____
Dr. Y.R. Shayan, Chair
Department of Electrical and Computer Engineering

_____ 20___

_____
Dr. Amir Asif, Dean,
Faculty of Engineering and Computer Science

Abstract

Connectivity Preservation in Multi-Agent
Systems using Model Predictive Control

Ahmed El-Hamamsy

Flocking of multi-agent systems is one of the basic behaviors in the field of control of multi-agent systems and it is an essential element of many real-life applications. Such systems under various network structures and environment modes have been extensively studied in the past decades. Navigation of agents in a leader-follower structure while operating in environments with obstacles is particularly challenging. One of the main challenges in flocking of multi-agent systems is to preserve connectivity. Gradient descent method is widely utilized to achieve this goal. But the main shortcoming of applying this method for the leader-follower structure is the need for continuous data transmission between agents and/or the preservation of a fixed connection topology. In this research, we propose an innovative model predictive controller based on a potential field that maintains the connectivity of a flock of agents in a leader-follower structure with dynamic topology. The agents navigate through an environment with obstacles that form a path leading to a certain target. Such a control technique avoids collisions of followers with each other without using any communication links while following their leader which navigates in the environment through potential functions for modelling the neighbors and obstacles. The potential field is dynamically updated by introducing weight variables in order to preserve connectivity among the followers as we assume only the leader knows the target position. The values of these weights are changed in real-time according to trajectories of the agents when the critical neighbors of each agent is determined. We compare the performance of our predictive-control based algorithm with other approaches. The results show that our algorithm causes the agents to reach the target in less time. However, our algorithm faces more deadlock cases when the agents go through relatively narrow paths. Due to the consideration of the input costs in our controller, the group of agents reaching the target faster does not necessarily result in the followers consuming more energy than the leader.

Acknowledgements

I would like to thank my supervisors Dr. Amir Aghdam and Dr. Farhad Aghili for their help, guidance, and patience during my research period. I am also grateful for my friends and colleagues for their helpful suggestions during the research meetings. Lastly, I cannot express enough gratitude for my family and teachers whom without their sacrifices and encouragement I would not have been able to reach this point.

Contents

List of Figures

List of Tables

List of Abbreviations

MAS	**M**ulti **A**gent **S**ystems
MPC	**M**odel **P**redictive **C**ontroller
PoF	**P**oint **o**f **F**ailure
UAV	**U**nmanned **A**erial **V**ehicle

Chapter 1

Introduction

In search for cheaper, more efficient, and more robust solutions for control problems across different fields, control of multi-agent systems has attracted researchers from vastly different fields in the past few decades. multi-agent systems (MAS) is a branch of control systems that utilizes groups of autonomous agents connected through a network operating in a specific environment in order to implement control algorithms that are able to achieve a specified task.

One of the most important advantages of MAS is their robustness with respect to a single point of failure (PoF) represented by a single complex agent, for example. Instead, having multiple simpler agents interact in the environment significantly reduces the risks since the malfunctioning of one agent does not necessarily jeopardize the whole mission. This can prove vital in operations that can benefit from taking risks such as space exploration missions.

An MAS can be described as a group of agents that function in a networked environment whose most important feature is their autonomous nature which allows the scalability of the system without increasing its complexity. Therefore, global control objectives can be reached using design strategies through distributed sensing, communication, computation, and control.

A significant aspect of MAS is their applicability in a plethora of fields including both technical and social sciences. Examples of small-scale MAS can be formation of unmanned aerial vehicles (UAVs), cooperative micro robots, coordination of autonomous cars, air transportation systems, and aerospace explorations while examples of large-scale MAS can be smart grids in a power system, traffic networks, sensor networks, biological systems, and social networks.

A good survey of the general framework and the details of the collective control of MAS can be found in books such as [3–10] and survey papers such as [1, 11–14]. The following section will review the related works in the literature.

1.1 Behaviors of Multi-agent Systems

The most important property of MAS is the emergence of complex behaviors on the collective level from very basic behaviors on the individual one. In this section, we discuss some of these basic collective behaviors in a brief manner.

The first collective behavior we discuss here is consensus. Consensus among multi-agents is considered to be the main pillar upon the rest of behaviors are built on. Consensus means the agreement between the agents on certain values or states that are related to the environment or to the agents themselves. Examples of these states can be the temperature of a room or the positions of an obstacle or the distance to a target.

Formation control, a behavior based on consensus, is where the agents move through space in certain shapes and in an ordered manner along a desired trajectory reference. In formation control, the main factor for its success is based on reaching certain positions, velocities, and maintaining inter-agent distances. The connectivity between the agents can be assumed but when connectivity is not assumed, it can be maintained by setting algorithms or requirements that aim to preserve the connectivity.

Flocking is a behavior that is inspired from nature where flocks of birds or schools of fish move together in harmony through space. Therefore, in MAS, agents are to move through space while following the three rules proposed by [15] as follows:

1. Flock Centering: each agent has an idea where the center or the majority of the agents are in the environment not to lose connection.
2. Collision Avoidance: each agent has to be able to avoid collisions either with natural elements in the environment or with other agents.
3. Velocity Matching: each agent needs to synchronise its velocity with the other agents in the flock in order to achieve the previous two behaviors.

A flock can have a leader, multiple leaders, a virtual leader i.e.: just a trajectory to go by, or it can have no leaders at all.

1.2 Components of Multiagent Systems

After going through some of the basic behaviors that multi-agents do, we now explain how they can interact with each other in an environment. Cooperative agents performing a common task require links among them whether these links are sensing links where agents have sensors to be able to locate other neighbors or objects in the environment or communication links for agents to share relevant information about a state in the environment or states concerning the agents themselves.

The nature of these links change significantly from one work to another based on the application needed and the physical structure of the agents themselves. We classify the possible structures of links between the agents as follows:

1. Undirected Link Structure

   This structure means that that the link between two agents goes in both directions, e.g., if

agent A can sense or communicate with agent B, then agent B can sense or communicate with agent A as well.

2. Directed Link Structure

   This structure means that the link between two agent does not necessarily go in both directions, e.g., if there is a directed link that goes from agent A to agent B, then agent B can sense or obtain information from agent A but agent A can not do the same unless there exists a directed link from agent A to agent B.

   A strongly connected directed graph is defined as a graph that has information able to be transmitted from one agent to any other agent contrary to a weakly connected directed graph. A directed graph can be weakly connected but can have a directed spanning tree where there is a directed path from one agent known as the root to any other agent.

3. Leader-Follower Structure

   This structure is used when the agents are classified into leaders and followers which can be based on the type of problem that the multi-agents need to solve. Usually, links that the leader has are only directed links going towards its followers and the links between the followers themselves can either be directed or undirected.

   A leader is usually used in tracking problems where the followers track the state of the leader to achieve consensus. The followers achieve tracking of an active leader i.e.: with a changing state and estimate this state when the direct measurement of states is not available and they have to be estimated based on their neighbors in [16] and [17]. Estimation of states is commonly studied in leader-follower topologies that need to be robust since the agents can be subject to measurement noise as in [18], model uncertainties as in [19] and [20], or delays as in [21].

An example of the relation between the agents in an undirected graph is shown in Figure 1.1 from [1] where agents 1 and 2 share their information while agent 6 only receives information from agent 5 without sending it to any agent, for example.

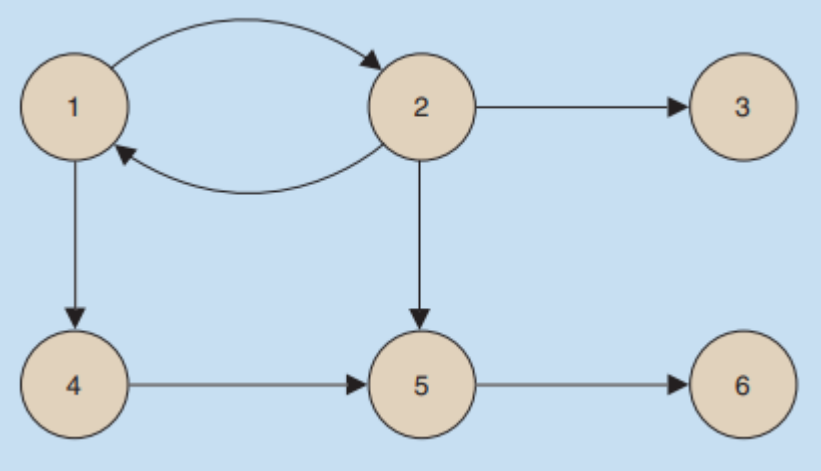

FIGURE 1.1: An example of a directed graph from [1]

Another method by which we can classify the links is by their dynamics that change based on the restrictions put, application required, and assumptions made regarding the environment and the agents as follows:

1. Fixed Topologies

   This means that the links between the agent remain the same throughout time. This topology is usually used in works where connectivity is assumed continuously or the loss of the links between the agents is not probable.

   In [22], velocity consensus between agents was achieved with the desired inter-agent distances while avoiding collisions with each other in a fixed topology. The authors in [23] show that consensus is achieved if the directed communication topology has a rooted directed spanning tree or the undirected communication topology is connected. Non-holonomic agents were studied in [24] where agents move towards the average positions of their neighbors under the effect of external disturbances using a distributed sliding mode control.

2. Switching Topologies

   This means that agents change their connections based on a finite known set of distinct graphs as time passes i.e.: for each time step, each agent can have a new set of agents

that it links with. The set of graphs should maintain the global connectivity where no agent will lose all links in a time step.

The authors in [23] show that consensus in switching networks with directed communication topology is reached if the union of the directed interaction graphs has a spanning tree frequently enough.

3. Time-Varying Topologies

   This type is the same as switching topologies except that the set of graphs is infinite in this case. The links that each agent has, however, do not change in the same manner as in switching networks but mostly the change is dependant on the trajectories that mobile agents have during the process. Time-varying or time-dependant graphs are heavily used in research works that study connectivity preservation.

   The condition for consensus in switching networks for undirected graphs is shown in [25] to be able to extend to time-varying topologies where consensus is achieved if there exists an infinite sequence of time intervals such that the union of the graphs is connected frequently enough as system evolves. Flocking behavior is studied in [26] and the agents are proven to have their velocity aligned as long as connectivity is maintained regardless of changes occurring in the network graph.

After looking into the links between agents and their dynamics, the dynamics of the agents themselves can be classified into:

1. Linear Homogeneous Agents

   which means that all the agents follow the same linear dynamics where $\dot{x}_k(t) = Ax_k(t) + Bu_k(t)$, x is the agent state at time t, u is the agent input at time t, k is the agent index, and A and B are constant matrices.

2. Linear Heterogeneous Agents

   which means that the agents do not necessarily have similar dynamics but they are still

linear in nature where in $\dot{x}_k(t) = A_k x_k(t) + B_k u_k(t)$, the constant matrices A_k and B_k are specific for each agent k.

The cooperation in a linear heterogeneous agent is different from synchronisation in homogeneous agents where consensus is needed. The agents should be able to regulate their outputs to cooperate while having different final trajectories which is known as the cooperative output regulation problem as in [27] and [28]. Robust approaches were developed in [29] and [30].

3. Nonlinear Dynamics
   where they can be either homogeneous as in $\dot{x}_k(t) = \phi(x_k(t)) + Bu_k(t)$ or heterogeneous as in $\dot{x}_k(t) = \phi_k(x_k(t)) + B_k u_k(t)$.

   One approach that is used when dealing with nonlinear dynamics is the virtual exosystem approach where the output of each agent is regulated individually to its reference then the references for all the agents are to reach consensus. Some of the papers that study the synchronisation of nonlinear agents using this approach are [31–35].

Figure 1.2 shows how the components of an MAS are related to each other.

The MAS can be classified into three categories based on the decision making strategy:

1. Centralized Control which means that all information, measurements, and calculations done by the agents are gathered by a main agent that determines the control actions that should be done by each agent.

2. Distributed Control which takes into consideration that having a central agent implies a single point of failure and distributes the decision across multiple central agents across all the other agents.

3. Decentralized Control which takes the distributed approach to its full and removes centralization completely where each agent is its own decision maker based on the information and measurements done gathered from the neighbors and the environment.

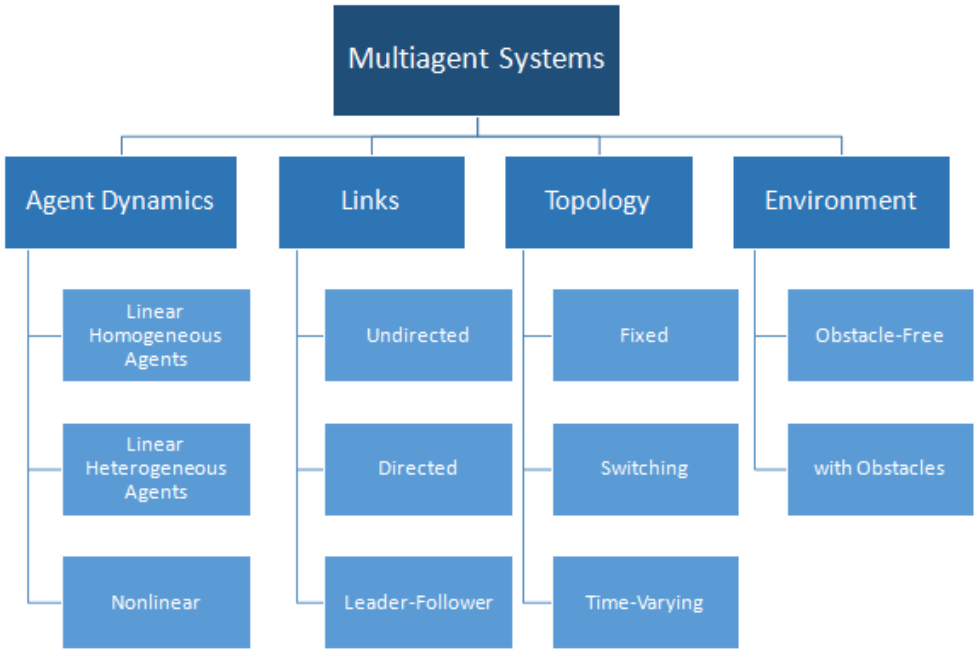

FIGURE 1.2: A hierarchy that shows the structure of an MAS based on its components

And for the environment in an MAS, it can have obstacles or be obstacle-free. The presence or the lack of presence of obstacles in an environment affects heavily how the agents interact with each other and the environment depending on the type of links the agents have since the connections between the agents can be based on either communication or sensory devices or both.

Despite the massive research efforts exerted in the field of MAS, it remains a broad area of research and in this section, we go through some of the challenges that we have in small-scale MAS such as navigation of unmanned aerial vehicles and coordination between autonomous cars which can be considered some of the most essential applications of cooperative control of MAS in the modern world and some of the challenges that appear to emerge when looking at previous literature are:

- The negligence of direct physical coupling between the agents in addition to the designed cooperative connections.

- Heavy dependence on communication to perform collective behaviors such as information exchange needed for distributed estimation or connectivity preserving algorithms.
- Addressing the synchronization problem of nonlinear and heterogeneous dynamics of MAS such as different unmanned aerial vehicles in a flight formation control or power stations in a smart grid with different load and supply characteristics.
- Investigation of collective behaviors of non-holonomic agents in a complicated network contrary to the usual modeling of agents as point masses without any input cost considerations or moving through obstacle-free spaces.

We can see that these challenges are mostly linked to real life situations and applications and addressing them is the necessary and logical step in this area of research.

1.3 Literature Review

In this thesis, we focus on applications that utilize the navigation of multiple agents in a leader-follower structure in an environment with obstacles without reliable communication links and where the agents are to conserve their energies. and since flocking of multi-agents is one of the basic applications of the field of MAS, it has been studied extensively in the past couple of decades. In this section, we review some of the literature that discuss the algorithms used in flocking in different situations in order to demonstrate how we were able to reach our point of interest in this work.

One of the most influential papers in this area is [36] as it presents a theoretical framework for the design and analysis of flocking in a distributed manner and divides the three main factors in the environment into agents which are the flock members (α- agents), the obstacles (β- agents), and the collective potential or objective (γ- agents) which is viewed in this paper as a moving rendez-vous point. A construction of the cost function or collective potentials based on these three types of agents, therefore, is provided systematically. It is to be noted that, although

the topology is time-varying, it only moves towards increasing connectivity by adding more links where connectivity is already assumed.

Since connectivity can not always be assumed in real life applications. The authors in [37] provide algorithms for both cases where connectivity is assumed and when it is not. In this paper, they study the distributed tracking problem with a virtual leader that moves with constant or varying velocity in fixed and switching networks and places a connectivity requirement based on potential fields in the second case. The virtual leader in this paper is a neighbor of a limited number of followers and can only perform communications and local interactions with this limited group of neighbors.

A lot of literature, as well as this thesis, focus on the connectivity maintenance aspect that is related to flocking, tracking, or navigation in the workspace. One of the earliest papers that worked on ensuring connectivity of the agents is [38] as it was necessarily needed while performing a certain global objective such as: rendezvous or formation control and this paper proposed assigning certain positive weights to the links between the agents that the Laplacian is built on but the graph of the agents is fixed, nonetheless, and the environment is free from obstacles.

Another paper that focused on the rendezvous problem in an obstacle-free environment without assuming connectivity is [39] where they have a virtual leader and new links keep getting added by constructing bounded potential functions that do not go to infinity. They showed that their algorithm works and rendezvous is achieved even if at least one agent has access to the information about the virtual leader.

The papers we have mentioned so far have assumed that the status of global connectivity is known for the agents, now, we start looking into some papers that utilize data transmission between agents as in [40] where a bounded controller is used to maintain the connectivity of the limited sensing and communications links between the agents while navigating through an environment with obstacles. The algorithm provided in this paper has its focus on connectivity preservation but other important aspects such as inter-agent collision are not considered.

The authors in [41] propose an algorithm that obtains a decentralized gradient controller that is based on an artificial potential field. The connectivity between the agents is quantified by the second smallest eigenvalue of the Laplacian graph which is estimated from information received from neighbors and the objective of the algorithm is to maximize this value which increases the connectivity.

The authors in [42] also use estimations of the algebraic connectivity to preserve connectivity but take into consideration bounded measurement errors of the information obtained from communication links between neighbors. In addition to that, the paper also introduces the concept of critical robots, which a disconnection from leads to the disconnection of the whole network, and limits the control actions to these critical robots to avoid unnecessary actions. The dynamics of the agents are single-integrator dynamics and the agents flock in an obstacled environment.

Till now, the topology in our discussion has either been fixed or increasing in links. The concept of deleting the communication links of the agents is introduced in [43] where agents flock in an obstacle-free environment so a discrete network topology controller is combined with a continuous controller that ensures velocity synchronization and collision avoidance that is based on an artificial potential field into a hybrid architecture.

In some papers, the leader-follower structure is implemented in the flocking of multi-agents where only the leaders in the network graph have the global knowledge in the environment whether in an obstacle-free environment as in [44] or in an obstacled one as in [45]. In [44], the agents estimate the flocking center by using a consensus algorithm that is implemented by utilizing data transmission through communication links. And in [45], the leaders steer the rest of the agents into desired formations to maneuver their path in an environment with obstacles.

An approach that is studied is the flocking of MAS without data transmission which we see in literature such as [46] which reaches a sub-optimal solution for leader-follower navigation in an obstacle-free environment utilizing a negative gradient controller based on potential fields and constraints the inputs of the agents to maintain connectivity. The authors in [2] build

on this paper and put explicit input constraints that are applied on all agents, the leader and the followers, for maintaining connectivity in a flock with dynamic topology while navigating through an obstacled environment.

1.4 Challenges and Contributions

Most of the previous literature reviewed have had one or more of the following limitations:

1. The consideration of an obstacle-free environment
2. The requirement of data transmission through communication links between the agents for estimation, auction algorithms, formation maneuver strategies, or otherwise.
3. Maintaining connectivity of the agents through either preserving a fixed network topology or increasing the links between the agents compared to the initial graph.
4. No consideration of the occurrence of inter-agent collisions
5. No explicit consideration of input constraints
6. Placing connectivity constraints on the leaders in a leader-follower structure

In this thesis, we propose a Model Predictive Controller (MPC) based on potential fields that maintains the connectivity of a flock of agents while navigating through an environment with obstacles in a leader-follower structure. The followers, in our case, which have limited sensing capabilities have to avoid collisions with each other and with the obstacles in the environment while following the leader which is the only agent that has access to the desired trajectory and moves with constant velocity with no constraint placed on it.

A contribution of our thesis is the introduction of weights of links between the agents into the construction of the potential functions that are used in the control algorithm used for the agents. By adding these weights, the agents can move into the general direction of the leader without

knowing which particular agent is actually the leader in our graph. Therefore, our problem can be described as a flocking problem in a leader-follower topology where the leader moves in a constant velocity through space with static obstacles towards a certain trajectory and the agents are to follow the leader while maintaining global connectivity and avoiding collisions with obstacles or with each other depending only on sensing links without any communication. The agents model their neighbors and the obstacles as weighted potential functions and decide their input for each time step based on a Model Predictive Controller (MPC) while taking into consideration their input costs as well.

1.5 Outline

This thesis is composed of four chapters. In Chapter 2, we describe the problem formulation, assumptions, and the rules used in this work along with the reasons for their choices. It follows by illustrating the results obtained in an environment free from obstacles. In Chapter 3, we expand our problem formulation and derive the control laws needed in an environment with obstacles, then, the results are compared with those reported in the literature. Finally, Chapter 4 presents the conclusions of this thesis and provides comments for future work.

Chapter 2

Connectivity Preservation in an Obstacle-Free Environment

This chapter formulates and solves the connectivity preservation problem in an obstacle-free environment. The topics to be discussed include: i) how the environment is structured, ii) the dynamics of the agents, iii) how the obstacles are modeled, iv) the conditions placed on the agents, and v) the assumptions needed. Next, we explain how our algorithm works and the reason for the choice of specific methods instead of other ones. Finally, we illustrate the simulation results based on the proposed control law and then we examine the results when some of the parameters are changed.

2.1 Problem Setting

2.1.1 Models of Agents

Suppose the number of agents that flock in the 2D workspace be N and their dynamics be described as:

$$x_i^{k+1} = x_i^k + u_i^k, \quad x_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}, u_i = \begin{bmatrix} u_x^i \\ u_y^i \end{bmatrix} \tag{2.1}$$

where $i \in V := \{1,2,...,N\}$ is the agent index and k is the time step. That means that the agents change their positions each time step based on the input signal u_i in the x and y axes. As can be seen, the dynamics of the agents in our system are discrete where the path of the agent is composed of line segments $L(x_i^k, x_i^{k+1})$. If the agents are described by the continuous model $\dot{x}_i(t) = u_i(t)$, then a fixed $u_i(t)$ between time steps k and $k+1$ will produce the same behavior.

2.1.2 Behavior of Agents

The agents in our system should display two fundamental behaviors:

1. Connectivity maintenance
2. Collision avoidance

First, we establish the conditions surrounding the agents to make sure that they avoid collisions and have other agents stay in their sensing ranges.

Agent i can avoid collisions with other agents if:

$$\|x_i - x_j\|_2 \quad \geqslant \quad D_c \quad \forall j \in V \setminus \{i\} \tag{2.2}$$

where D_c is a constant value based on the physical structure of the agents where the agents are not assumed as point masses.

Agent i can sense other agents if:

$$\|x_j - x_i\|_2 \quad \leqslant \quad D_s \tag{2.3}$$

Therefore, an agent j is sensed by agent i if it falls into its sensing range and is not colliding with it. In the following section, we will show how the sensing will be affected if there exists other agents between agents i and j.

2.1.3 Dynamics of the Network Topology

By establishing these conditions, we can define our sensing network graph $G_s := (V, E_s(x^k))$, where $E_s(x^k)$ is the set of edges of the sensing graph at time k and construct our sensing Laplacian matrix $L_s(x^k)$ where, for $i \neq j$,

$$s_{ij} := \begin{cases} -1 & \text{if (2.2) and (2.3) hold} \\ 0 & \text{Otherwise,} \end{cases} \tag{2.4}$$

and

$$s_{ii} = -\sum_{j \in S_i^\sigma} s_{ij} \tag{2.5}$$

where $S_i(x^k) := \{j | (i,j) \in E_s(x^k)\}$ is defined as the set of agents that agent i can sense.

If the network sensing graph G_s is connected, each follower has a path going to the leader with a maximum length of $(N-1)D_s$. Trying to maintain the connectivity of our network just through guaranteeing the preservation of G_s can be unreliable since connecting edges between agents can be lost by having very minor movements or changes in the positions of the agents. Therefore, we aim to preserve connectivity of $G_n(x^k) := (V, E_n(x^k))$ where $E_n(x^k) := \{(i,j) \in E_s(x^k) | \|x_i^k - x_j^k\|_2 \leqslant D_n\}$ and $D_c \leqslant D_n \leqslant D_s$ which is a subgraph of the sensing graph G_s and

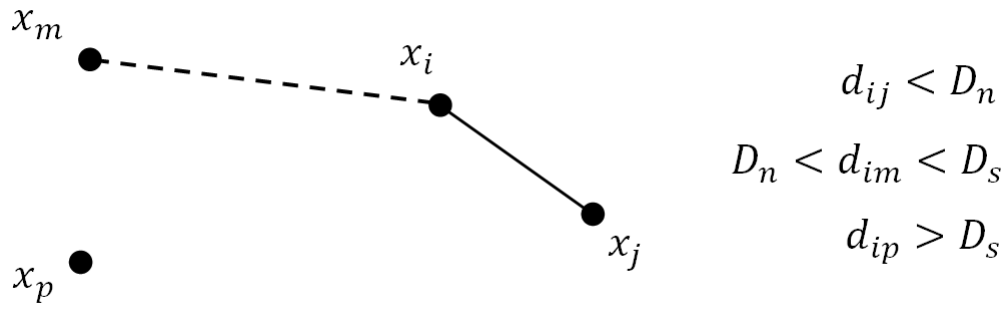

FIGURE 2.1: The method of classification of agents according to agent i

the neighbor Laplacian matrix $L_n(x^k)$ is, for $i \neq j$,

$$l_{ij} := \begin{cases} -1 & \text{if } s_{ij} = -1 \text{ and } \|x_i^k - x_j^k\|_2 \leqslant D_n \\ 0 & \text{Otherwise,} \end{cases} \tag{2.6}$$

and

$$l_{ii} = -\sum_{j \in N_i} l_{ij} \tag{2.7}$$

where $N_i := \{j | (i,j) \in E_n(x^k)\}$ is defined as the set of agents that are neighbors to agent i. The classification of the different relationships that agent i can have with other agents is shown in Figure 2.1 where agent $j \in E_s(x^k) \cap E_n(x^k)$, agent $m \in E_s(x^k)$ and $m \notin E_n(x^k)$, and agent $p \notin E_s(x^k) \cup E_n(x^k)$

Assumption 2.1: The leader is connected initially at ($k = 0$) to at least one follower.

Having this assumption, commonly, lots of papers aim to keep increasing the links between the agents as the system evolves, however, this can prove impractical in our problem since the leader agent is unknown to the followers and it is not bounded by the same desired behaviors imposed on the followers. Therefore, the followers should aim at preserving the connectivity with all of their neighbors. As the number of neighbors increase for a follower, the ability to find a solution gets harder and harder until the agent reaches a point where it can no longer move without losing its connection with at least one of its neighbors which is defined as a

deadlock case in our problem. Therefore, to be able to navigate narrow spaces in our environment and maintain connectivity of the graph at the same time, we need to deactivate the links that hinder the navigation process and keep the other links present in $E_n(x^k)$, so the set of neighbors to be preserved is defined as $N_i^{\sigma}(x^k) := \{j \in N_i(x^k) | \sigma_{ij}^k = 1\}$ and the following subgraph is defined as $G_{\sigma}(x^k) := (V, E_{\sigma}(x^k))$ where $E_{\sigma}(x^k) := \{(i,j) \in E_n(x^k) | \sigma_{ij}^k = 1\}$ where σ_{ij}^k is a systematic indicator function defined as

$$\sigma_{ij}^k := \begin{cases} 0 & \text{if edge } (i,j) \text{ is deactivated} \\ 1 & \text{otherwise} \end{cases}$$

We have two cases for link deactivation in narrow spaces:

1. For the set of triplets (i,j,m) illustrated in the left side of Figure 2.2 where $(i,j) \in E_n$, $(j,m) \in E_n$, and $(m,i) \in E_n$, the agent i does not include the edge $(i,j) \in E_n$ in E_{σ} if

$$\|\Phi(x_{mi}, x_{ji}\|_2 < D_c \sin(\pi/3) \tag{2.8}$$

$$\sin(\alpha_m) > 0, \quad \alpha_i, \alpha_j < \pi/2, \tag{2.9}$$

   which means that agent m is about to obstruct the vision between agent i and agent j, therefore, instead of having a direct sensing link between agents i and j, this link is removed and agent m becomes the connecting node that links between these two agents.

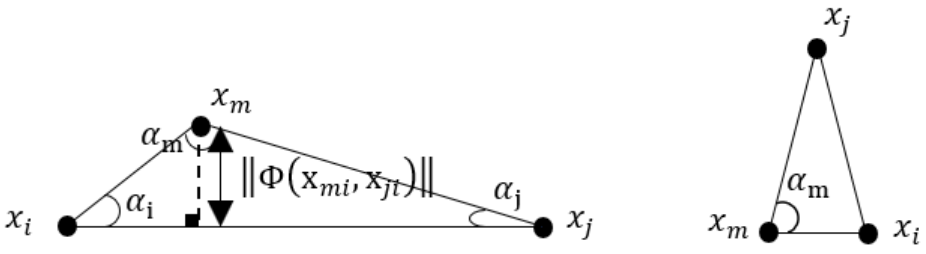

FIGURE 2.2: An illustration of the two conditions from left to right

2. For the set of triplets (i, j, m) illustrated in the right side of Figure 2.2 where $(i, j) \in E_n$, $(j, m) \in E_n$, and $(m, i) \in E_n$, the agent i does not include the edge $(i, j) \in E_n$ in E_σ if

$$\|x_{ij}\|_2 = \|x_{jm}\|_2 = D_n \tag{2.10}$$

$$\|x_{mi}\|_2 = D_c \tag{2.11}$$

$$\sin(\alpha_m) > 0, \tag{2.12}$$

which means that agent i and agent m are about to collide with each other and lose their connection to agent j, therefore, agent i drops its sensing link with agent j to avoid collision with agent m and allow it to maintain connectivity between itself and agent j, and hence agent m becomes the connecting node between agents i and j.

N.B.: The conditions for the second set of triplets allow for a certain range of tolerance because it is not practical to reach exact equality for these conditions, so, instead, we have:

$$D_n - \varepsilon_n \leqslant \|x_{ij}\|_2 \leqslant D_n \tag{2.13}$$

$$D_n - \varepsilon_n \leqslant \|x_{jm}\|_2 \leqslant D_n \tag{2.14}$$

$$D_c \leqslant \|x_{mi}\|_2 \leqslant D_c + \varepsilon_c, \tag{2.15}$$

where ε_n and ε_c are small positive numbers.

Assumption 2.2: The agents in the system have no measurement errors, therefore, if any of the two conditions for dropping links is fulfilled for one agent, it should be fulfilled for the other, i.e: if agent i drops its link with agent j at time k, then agent j drops its link with agent i at the same time step as well without any need for communication.

In order to ensure the connectivity of our sensing network after applying the two conditions of dropping links mentioned above, we mention the theorem that [2] used to prove the connectivity of $G_\sigma(x^k)$ and the proof of this theorem is presented there as well.

Theorem 2.1 [2]: Suppose that the following assumptions hold:

1. $G_n(x^k)$ is connected,
2. All agents satisfy (2.2),
3. The constraint in (2.8) is satisfied,
4. π is not an integer multiple of $\sin^{-1}(D_c/2D_n)$,

then $G_\sigma(x^k)$ obtained by the rule of σ_{ij}^k is connected. This means that σ_{ij} is determined such that if no edge is lost in G_σ, the connectivity of G_n is preserved.

N.B: The rule is decentralized and does not require data transmission between agents and multiple links of G_σ are deleted at the same time.

In [2], the leader of the agents in the network holds the same objectives that the followers do, so it can not move with an input that will cause connectivity with its followers to terminate. This is a constriction that is placed on the leader in this case and is represented explicitly by having an input constraint u_max that can change every time step but still has an upper bound based on the physical parameters of the agents in the system.
Such a constraint can be considered as unrealistic and, instead, the leader should move freely in the environment without considering connectivity preservation with the followers and the followers should be solely the ones with these constraints.

In our problem, the leader moves in the environment towards its designated target with the same input magnitude for each time step where the followers need to keep moving with the leader without knowing either the destination target or which agent is actually the leader.

Finally, this problem can be summarized as follows: A group of autonomous agents are to move in an environment without obstacles that form a path leading towards a target only known to the leader of the group. The followers have to maintain connectivity with their neighbors and, by extension, with the leader and form connections with new agents found along the way

while taking collision avoidance into consideration in a dynamic network sensing topology where links can be added or removed under certain scenarios and without any communication present between the agents.

2.2 Control Algorithm

After we have explored the foundations of our system and its modeling, in this section, we explain how the agents should interact with the environment in order to achieve the desired behaviors explained previously. Since the dynamics of the agents are discrete, an agent moves through space in discrete inputs with fixed time steps, therefore our objective is obtain the input for each time step that preserves connectivity and avoids collisions.

2.2.1 Modeling of the Potential Field

One of the most popular methods in literature to reach the desired behaviors is to model them as functions where each agent has a cost function that it tries to minimize by choosing the suitable input. The cost function is a potential field comprised of the sum of the potential functions that lead us to the behavioral objectives where each potential function focuses on one aspect of the behaviors.

In our algorithm, the agents are subjected to the potential field

$$U(x_i) := \sum_{j \in N_i^\sigma} c_1 f_1 + c_2 f_2 \tag{2.16}$$

that aims for connectivity maintenance and collision avoidance represented by its two functions. These two components will be explained in this section individually.

The first component f_1 takes into account the desired value of the relative distance to neighbors $j \in N_i^\sigma$ as agent i should maintain a distance value with its neighbors that maintains the

connectivity and prevents collisions with them at the same time. Therefore, the functions is defined as:

$$f_1(\|x_{ij}\|_2) := w_{ij}(\|x_{ij}\|_2 - D_r)^2 \tag{2.17}$$

where $\|x_{ij}\|_2 := \|x_i - x_j\|_2$, and w_{ij} is a variable that determines the weight of agent j assigned by agent i, meaning, neighbors of agent i do not necessarily hold the same value or cost when their relative distances to agent i are equal.

Since we have removed the constraint of connectivity preservation placed on the leader which, for the most part, will result in the leader moving away from its neighbors to reach the target, we needed to come up with a method for the followers to identify the leader or agents which follow the leader. Therefore, to preserve connectivity, a follower keeps increasing the weight of the neighbors that keep moving away from it relative to the other neighbors that do not, so $if\ d_{ij}^{k+1} > d_{ij}^{k}$:

$$w_{ij}^{k+1} := w_{ij}^{k} + \frac{(w_{ii}^{k} - 1)}{w_{ii}^{k}} \tag{2.18}$$

$$w_{\substack{im \\ m \neq j}}^{k+1} := w_{im}^{k} - \frac{1}{w_{ii}^{k}} \tag{2.19}$$

The rationale behind (2.18) and (2.19) is increasing the weight of one neighbor while decreasing the weight of all the other ones so that the total weight assigned by agent i to its neighbors remains constant. The sum of weights for agent i is defined as:

$$w_{ii} = \sum_{j \in N_i^{\sigma}} w_{ij} := |N_i^{\sigma}| \tag{2.20}$$

where $|N_i^{\sigma}|$ is the number of neighbors of agent i. The reason for keeping the total weight assigned for each agent constant is to keep the cost of function f_1 relatively the same as function f_2. Therefore, the only means to prioritize one of them over the other is through the gain values c_1 and c_2.

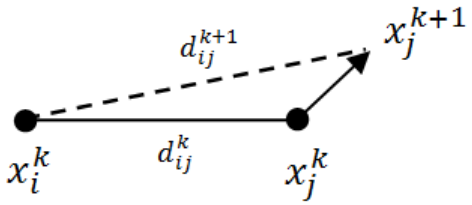

FIGURE 2.3: Agent j moves away from agent i

Assumption 2.3: Each agent stores the values of the positions of its neighbors for a certain number of the previous time steps.

Figure 2.3 is an illustration of how agent i monitors the way its neighbors move. This method of identifying the general direction to which the leader moves can be considered intuitive because if the leader were to reach its target, it would need to keep moving towards it regardless of the presence of the followers near it or not. Therefore, if an agent keeps moving away from its neighbors for a relatively long period of time, it means that it is either following another agent that is moving away from it or it is the leader going towards the target.

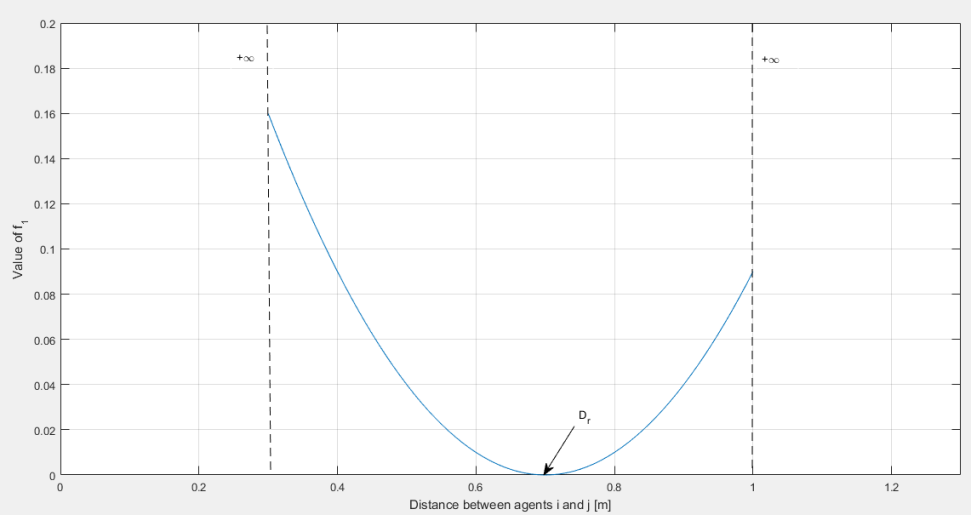

FIGURE 2.4: Plot of the potential function f_1 which shows its domain and fundamental parameters

The weight term added to function f_1 accomplishes our objective because even if two neighbors of agent i have the same distance from it, agent i will prioritize the connectivity with the neighbor with the larger weight because this agent keeps moving away in pursuit of either the target in case of the leader or another neighbor with a higher weight in case of the followers.

The function f_1 is modeled as a quadratic function whose minimum value is at D_r and increases in both directions to prevent collision and preserve connectivity as shown in Figure 2.4. We can see that function f_1 takes effect only between the collision distance D_c and the neighborhood distance D_n and other than that the value of the function goes to infinity as either collision occurs or neighbors are lost for the agent.

The second component f_2 is introduced to make the group more cohesive as long as it does not affect the functional behaviors of those agents. For example, assume agent i senses agent p, then it is not necessary that they are neighbors, but they also need to have a certain threshold distance of D_n between them for them to be considered neighbors. So, the function f_2 is presented where agent p is not a neighbor for agent i and agent i has little or no forces in terms of connectivity preservation or collision avoidance that are pushing it away from agent p. Therefore, in general, agent i moves closer to agents whose relative distances to it is larger than D_n and smaller than D_s at time step k. The function f_2 is defined as follows:

$$f_2(\|x_{ij}\|_2) := (\|x_{ij}\|_2 - D_n)^2 \quad \text{if} \quad D_s \geqslant \|x_{ij}\|_2 > D_n \tag{2.21}$$

The function f_2 is modeled as a quadratic function whose minimum value is at D_n because, as the distance goes smaller than that, function f_2 ceases to have an effect regarding this particular agent and function f_1 becomes the main force affecting in regards to connectivity preservation. Unlike function f_1, function f_2 does not go to infinity if the domain goes beyond the sensing distance D_s as it is not a crucial part of the connectivity preservation behavior of the agents to keep sensing agents that are not within its neighborhood set as shown in Figure 2.5.

Since the two functions that constitute our potential field have ranges of different values, we

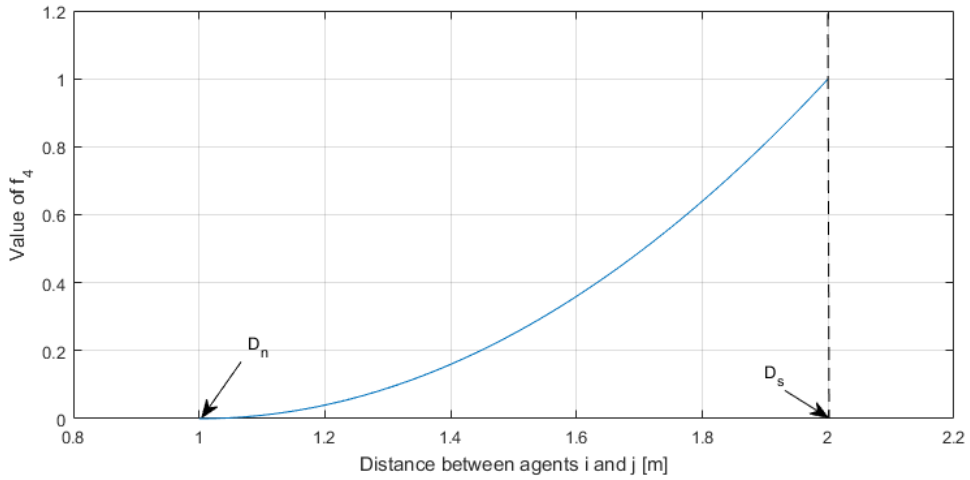

FIGURE 2.5: Plot of the potential function f_2 which shows its domain and fundamental parameters

need to re-scale them so that they both have the same values for their minimums and maximums and the same influence on the agent as they go towards their extremes.

If we desire to change the influence of one of the functions in relation to the other one, we can do that by changing the values of c_1 and c_2 of $U(x_i)$ in (2.16).

2.2.2 Obtaining the Input of the Agents

After modeling the potential field, utilizing it to acquire the inputs for the agents is the next step logically. An algorithm that is used commonly to do that is the negative gradient controller. The negative gradient controller or gradient descent, as defined in [47], is an optimization algorithm which is classified as a black-box optimizer and is used to minimize a cost function $J(x)$ by updating the states in the opposite direction of its gradient $\nabla_x J(x)$. This iteratively reduces the value of the cost function till it reaches a value that does not change and that is then considered the minimum of the function and the corresponding states as the minimizers.

There are three variants of gradient descent based on how much data is used to compute the value of the cost function at a time:

1. Batch gradient descent: the entire state or input set is computed for the whole mission at one time which can be considered as offline optimization.

2. Stochastic gradient descent: the input is computed for each time step after updating the states and parameters of the environment. This is considered as online optimization and the most popular variant used in the literature dealing with navigation of multiple agents.

3. Mini-batch gradient descent: this variant utilizes both of the previous variants and updates the states and parameters of the environment every mini-batch of time steps instead of each time step.

Although gradient descent algorithm is used extensively in papers that model the environment as a potential field to navigate as in [11, 39, 41, 42, 46], it has the following shortcomings when it comes to some of the desired behaviors we want our agents to express:

1. Not considering the cost of the inputs needed to optimize the states of the agents

2. The difficulty of minimizing highly non-convex cost functions with local minimums and saddle points

Another alternative controller that is used in algorithms that utilize potential function models is the model predictive controller (MPC) despite not being used extensively in MAS navigation. In an MPC, at time step k, the behavior of the agents is considered and the cost function $J(x)$ is computed over a horizon p, and the inputs are calculated responding to the variables predicted and aiming to minimize the cost function. After implementing the input for time step k, the whole operation repeats for time step $k+1$. As mentioned in [48], minimizing the cost function can be based on the presence of constraints to the input or the states or it can be unconstrained.

In literature, MPC was used to solve control problems such as path planning of autonomous vehicles in [49], formation control in [50] and [51]. Decentralized MPC (DMPC) is a type of MPC controller which considers each agent as a decoupled subsystem that solves a local sub-problem such as in [52] where it is used to control Unmanned Aerial Vehicles (UAVs)

and avoid collisions with each other. [53] used DMPC to control two UAV teams to encircle movable or stationary targets.

A small comparison is presented in table 2.1:

Gradient Descent vs. Model Predictive Controller		
Type of Controller	Gradient Descent	Model Predictive Controller
Dynamics	can be used to solve linear and non-linear systems	can be used to solve both types of systems as well
Cost Function	limited to differentiable functions and can not solve non-differentiable ones	explicit MPCs can solve piecewise affine functions
Horizon	the horizon of the data is based on one of the three variants explained above	the MPC works in a variable finite-time horizon
Input	the input magnitude is not usually explicitly considered	the input cost can separately be considered
States	constraints can be added by projecting their sets	can deal with both constrained and unconstrained state sets

TABLE 2.1: Table to illustrate the similarities and differences between the two controllers

We can see that the model predictive controller has wider scopes that it can be used for in terms of the cost function, horizon chosen, and input and state constraints. For this reason, we utilize the model predictive controller in our algorithm to put a cost to the inputs of the agents without needing to put an explicit input bound for connectivity maintenance which is already covered by our model of potential functions but, instead, the input cost is used to ensure that the agents do not unnecessarily take big leaps in their movements through space that will consume lots of energy and do not, in return, have a big impact on the connectivity of the agents which is more in line with real-life restrictions. Therefore, our problem can be defined as follows:

$$\text{minimize} \qquad J = U(x_i) + u_i^T R u_i \tag{2.22}$$

$$\text{subject to} \qquad x_i^{k+1} = x_i^k + u_i^k \tag{2.23}$$

where J is the cost function and R is a positive-definite matrix. The big $\mathcal{O}$ notation shows that the computational complexity of this solution is of order $\mathcal{O}(N^3)$ where N is the number of agents.

In certain cases, our algorithm fails to reach a solution where the agents experience deadlock and can no longer continue the mission without losing connectivity, colliding with obstacles, or colliding with each other. Therefore, we need to inspect these cases, as well as, the effect of changing the weight values of the links between the agents.

Theorem 2.2: If agent i is connected to agent j at time k, then $\exists u_i \in \mathbb{R}^2$ for which the agent i stays connected and does not collide with agent j at time $k+1$ if:

1. the weight assignment of agent j by agent i is done in such a way to satisfy the following bounds

$$0 \leqslant w_{ij} \leqslant |N_i^\sigma|$$

2. The value of $c_1 f_1$ in $U(x_i)$ does not go to infinity and the cost for each neighbor lies in the following range

$$c_1 w_{im}^k (\|x_{im}^k\|_2 - D_r)^2 \leqslant 1, \quad m \in N_i^\sigma$$

Before going into the proof of this theorem, we need to examine the two conditions mentioned above. The first condition is based on our choice of the weight values in function f_1 which will be explained later in the proof. For the second condition, it depends on the exact circumstances present in our environment during time step k, therefore, it does not always hold which leads to a deadlock case where the agent can not find a solution or an input that will make it move without violating at least one of the conditions of connectivity preservation or collision avoidance.

We check the cases where any input for agent i will lead function f_1 to infinity. In Figure 2.6, we can see an example of a deadlock case for agent i due to the lack of a solution that keeps

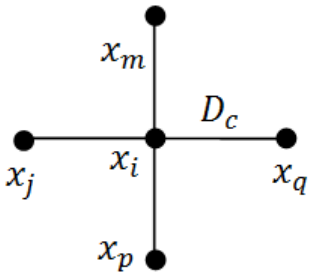

FIGURE 2.6: Agent i is surrounded by four neighbors with equal distances from it of value D_c

function f_1 within range. To prove that it is a deadlock case, we show a case where agent i needs to move away from agents p and q for example, we will have to know what angle the projectile of the agent can make with the other agents in its neighbourhood set without colliding with them. In the triangle shown in Figure 2.7, we should have $b > D_c$ if agent i is to have moved away from its neighbors and since the cosine law can be written as $b^2 = a^2 + D_c^2 - 2aD_c\cos\phi$, we can have:

$$b^2 > D_c^2 \quad \text{by squaring both sides} \tag{2.24}$$

$$a^2 + D_c^2 - 2aD_c\cos\phi > D_c^2 \quad \text{from the cosine law} \tag{2.25}$$

$$a^2 - 2aD_c\cos\phi > 0 \tag{2.26}$$

$$a(a - 2D_c\cos\phi) > 0 \tag{2.27}$$

The solution of this inequality can be either ($a < 0$) meaning vector a goes in the opposite direction or ($a > 2D_c\cos\phi$). However, to preserve convexity, this solution is only valid if $\phi > \pi/2$. This means that agent i will collide with agents m or q because if π is less than that, this solution is going to cause the collision with agents p, q, or both which proves the deadlock in this example. A solution that satisfies the inequality but does not preserve convexity will require that a be greater than a certain positive value, but it is not enough for the agent to have a safe position or destination in the next time step $k+1$, it also needs to go there without colliding as well.

For the other aspect of function f_1, we check the condition for connectivity maintenance for

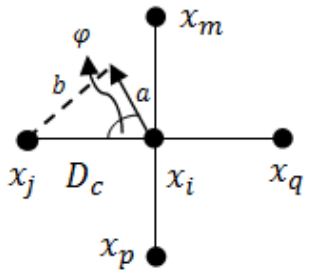

FIGURE 2.7: Investigating the trajectory of agent i while trying to move away from its neighbors

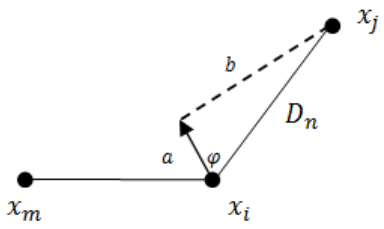

FIGURE 2.8: Investigating the trajectory of agent i while trying to maintain connection with its two neighbors

agent i. In order to maintain connectivity in the illustration shown in Figure 2.8, we should have $b < D_n$ and since the cosine law can be written as $b^2 = a^2 + D_n^2 - 2aD_n\cos\phi$, we can have:

$$b^2 < D_n^2 \quad \text{by squaring both sides} \tag{2.28}$$

$$a^2 + D_n^2 - 2aD_n\cos\phi < D_n^2 \quad \text{from the cosine law} \tag{2.29}$$

$$a^2 - 2aD_n\cos\phi < 0 \tag{2.30}$$

$$a(a - 2D_n\cos\phi) < 0 \tag{2.31}$$

The solution of this inequality can be either $(a > 0)$ or $(a < 2D_n\cos\phi)$, therefore, this leads to deadlock when $\phi = \pi$ because a will always be bigger than zero as long as ϕ is less than 180 degrees i.e: the three agents are not colinear which is an intuitive result.

After going through the conditions of the theorem, now, we present the proof. The next steps are focusing on inter-agent collision avoidance or connectivity preservation because of our choice of f_1 as a symmetric function. We will go with the former in this proof.

Considering agent j as the neighbor agent i wants to avoid collision with and based on the second condition, we want to have:

$$c_1 f_1(\|x_{ij}^k\|_2) \leqslant 1 \tag{2.32}$$

$$c_1 w_{ij}^k(\|x_{ij}^k\|_2 - D_r)^2 \leqslant 1 \quad \text{from (2.17)} \tag{2.33}$$

and applying the same inequality from the perspective of agent j with agent i as its neighbor, we get:

$$c_1 w_{ji}^k(\|x_{ji}^k\|_2 - D_r)^2 \leqslant 1 \tag{2.34}$$

Since $\|x_{ij}^k\|_2 = \|x_{ji}^k\|_2$ based on Assumption 2.2, we get:

$$c_1(\|x_{ij}^k\|_2 - D_r)^2(w_{ij}^k + w_{ji}^k) \leqslant 2 \tag{2.35}$$

by adding (2.33) and (2.34).
We define $c_1 := \frac{1}{(N-1)(D_c-D_r)^2}$ to get:

$$\left(\frac{1}{(N-1)(D_c-D_r)^2}\right)(\|x_{ij}^k\|_2 - D_r)^2(w_{ij}^k + w_{ji}^k) \leqslant 2 \tag{2.36}$$

Since $max(\|x_{ij}^k\|_2 - D_r)^2 = (D_c - D_r)^2$, we can rewrite (2.36) as the worst case where agent j is the closest to agent i without colliding and, in this case, we seek the inequality in (2.36) to be in the following forms:

$$\left(\frac{1}{(N-1)(D_c-D_r)^2}\right)(D_c - D_r)^2(w_{ij}^k + w_{ji}^k) \leqslant 2 \tag{2.37}$$

$$\frac{1}{(N-1)}(w_{ij}^k + w_{ji}^k) \leqslant 2 \tag{2.38}$$

$$w_{ij}^k \leqslant 2(N-1) - w_{ji}^k \tag{2.39}$$

and since $0 \leqslant w_{ji} \leqslant |N_j^\sigma|$ based on generalizing the first condition of the theorem to all agents, we rerwite (2.39) as:

$$w_{ij}^k \leqslant 2(N-1) - |N_j^\sigma| \tag{2.40}$$

where $|N_j^\sigma|$ is the number of neighbors of agent j. Since $1 \leqslant |N_j^\sigma| \leqslant N-1$ for a connected network, we get:

$$w_{ij}^k \leqslant N-1 \quad \text{for } |N_j^\sigma| \text{ equals to } N-1 \tag{2.41}$$

since $min\ (2(N-1) - |N_j^\sigma|) = N-1$. This shows that even for the highest possible value w_{ij}^k can have, The inequality is not violated and we the weight agent i assigns for agent j will always fall within a range that will maintain the connectivity between the two agents as long as the two conditions in the theorem hold.

By applying the second condition of the theorem for the entire neighborhood set of agent i, we will have:

$$\sum_{j \in N_i^\sigma} c_1 f_1(\|x_{ij}^k\|_2) \leqslant |N_i^\sigma| - 1 \tag{2.42}$$

2.3 Simulation Results

To explore the effectiveness of the proposed algorithms explained in the previous section, we run various simulations in Matlab^{TM} environment using the *fmincon* function. We use the default algorithm used in the *fmincon* function which is the interior-point method as it satisfies our requirements in this work regarding the computation time and the number of agents.

The parameters in our formulation can be classified into two types:

1. Physical parameters that can not be changed unless we change the physical structure of the agents or the environment

2. Parameters that we choose based on our models and algorithms

The parameters that fall into the first category are:

(a) The maximum sensing range (D_s) of the agents which depends on their physical sensors.

(b) The minimum distance between the agents and each other (D_c) at which collision occurs which depends on the physical structure of the agents.

(c) The minimum distance between the line of sight between two agents and the obstacles (D_l) at which the vision between the agents is obstructed. This parameter depends on the placement of the sensors on the agents and the structure of the obstacles in the environment.

We have $D_s = 2m$, $D_c = 0.3m$, and $D_l = 0.05m$ which are the same values as in [2]. And the parameters that fall into the second category are:

(a) The number of agents (N)

(b) The maximum distance between neighboring agents (D_n)

(c) The desired distance between the agents and each other (D_r)

(d) The tolerance values ε_n and ε_c assigned to fulfill the second condition of link deactivation

(e) The gain values c_1 and c_2 of the functions in the potential field and the value of the positive definite matrix R for the cost input

We choose $N = 5$, $D_n = 1m$, $D_r = 0.7m$, $\varepsilon_n = \varepsilon_c = 0.1m$, $R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $c_1 = 1.04$, and $c_2 = 0.16$ to have the functions are equal in terms of their maximum costs.

We show in Figure 2.9 how the agents navigate the environment based on our algorithm where the leader goes through a certain trajectory towards a designated target. We see how links keep increasing between the agents and how new links are formed with other agents as the agents sense each other.

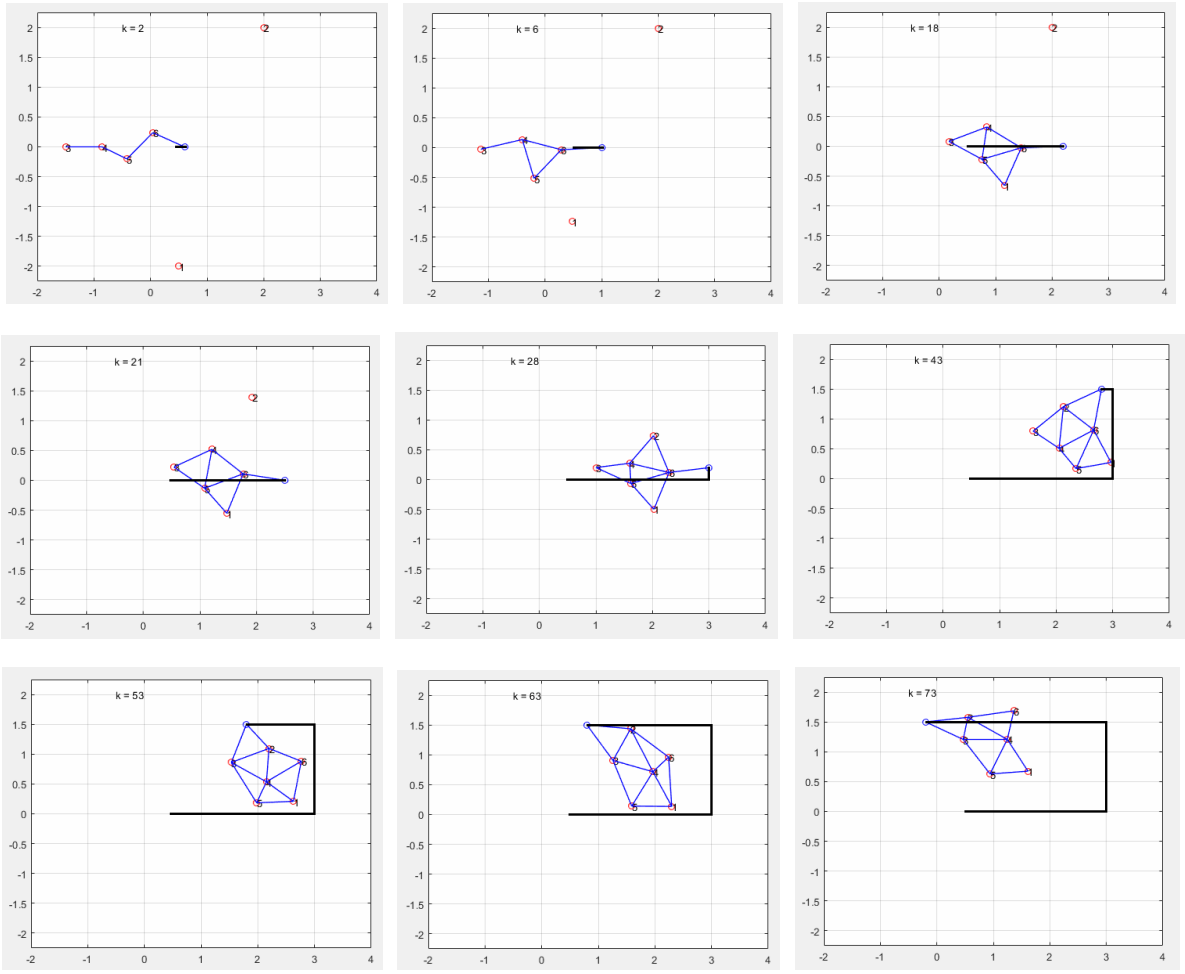

FIGURE 2.9: Snapshots for the navigation algorithm with the leader moving at constant speed of $u_N = 0.1m$ per time step

We show in Figure 2.10 how the navigation process can be hindered where the system eventually faces a deadlock case where the agents can no longer resume navigation without an agent losing connectivity with at least one of its followers. We use the the second smallest eigenvalue, known as the Fiedler value, of the Laplacian matrix L_n defined previously as a measure of connectivity and we show, in Figure 2.11(a), an example of the connectivity status of the agents while flocking. In Figure 2.11(b), the input magnitude of one of the followers is shown where the input of the leader is at $u_N = 0.1m$ per time step.

In summary, we can see from simulations that despite not constraining the leader with any connectivity-preserving behavior, the followers are able to identify the trajectory that the leader

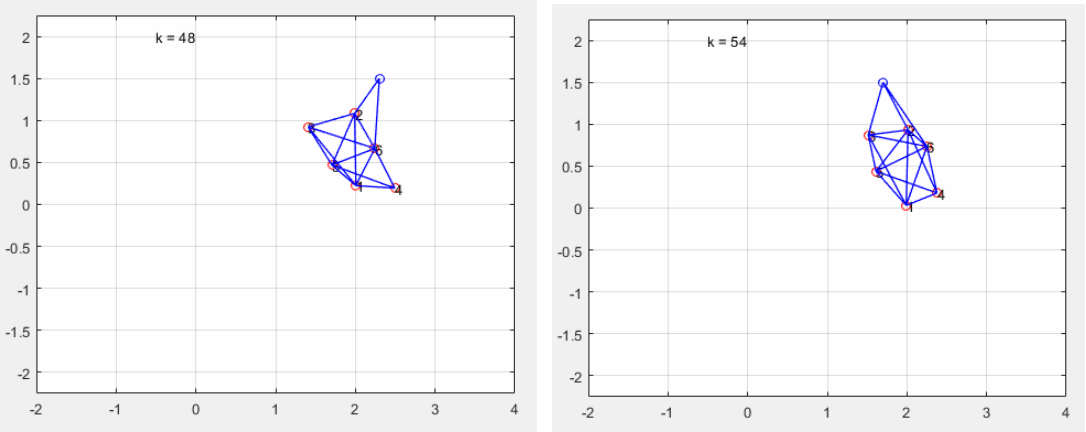

FIGURE 2.10: An illustration of the failure of navigation without the implementation of link deactivation

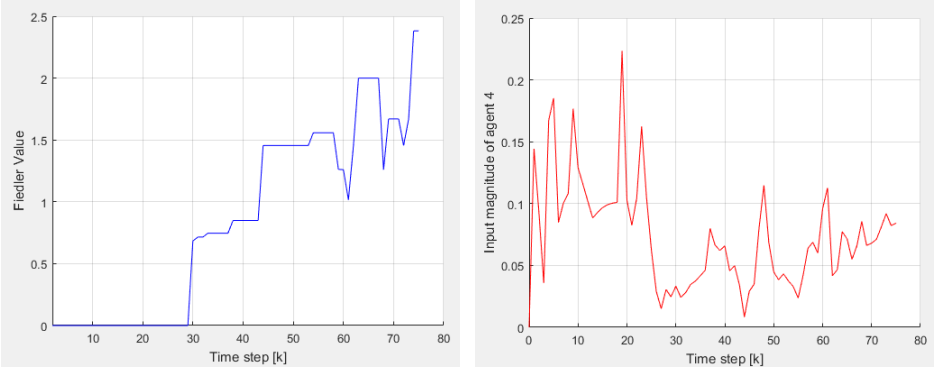

FIGURE 2.11: (a) The Fiedler value of the neighborhood connectivity matrix and (b) the input magnitude of an agent during the simulation run

takes to reach the target. It is also shown that link deactivation is necessary in order to avoid hindering the flocking process of the agents where extra links that are not needed are removed.

Chapter 3

Connectivity Preservation in Obstacle Environments

In this chapter, we expand our control algorithm to solve the navigation problem in environments with obstacles. First, we explain how the problem formulation changes by discussing how the environment is structured, the dynamics of the agents and the obstacles are modeled. Next, we explain how our algorithm is modified and, finally, we show our simulation results and compare them to previous works.

3.1 Problem Setting

3.1.1 Models of Agents and Obstacles

Let the number of agents that flock in the 2D workspace be N and their dynamics be described as:

$$x_i^{k+1} = x_i^k + u_i^k, \quad x_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}, u_i = \begin{bmatrix} u_x^i \\ u_y^i \end{bmatrix} \tag{3.1}$$

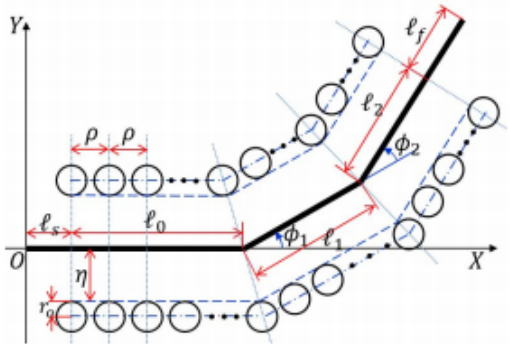

FIGURE 3.1: Forming a path made of obstacles with varying angles and lengths from [2]

where $i \in V := \{1, 2, ..., N\}$ is the agent index and k is the time step. We can see that the agents have the same dynamics as in Chapter 2, as for the obstacles present in the environment, they are modeled as static circles with equal fixed radii where a chain of consecutive circles can form a path in the workspace as shown in Figure 3.1.

The obstacles can be modeled as to resemble a couple of straight lines, or curves, or any other shape simply by changing where and how the circles are placed. In our problem formulation, the obstacles form a path that can have a certain width, turns, and length. The reason that circles are specifically chosen instead of other geometric shapes is that the agents in our system can calculate basic rules of geometry concerning circular shapes to achieve obstacle avoidance and connectivity preservation. This point will be further demonstrated when explaining the function models. Therefore, this does not mean that our algorithm is limited to only circular obstacles and obstacle modeling can be done in various ways based on the sensing capabilities of the agents in those systems.

3.1.2 Behavior of Agents

The agents in our system should display three fundamental behaviors:

1. Connectivity maintenance

2. Collision avoidance

3. Obstacle avoidance

First, we establish the conditions surrounding the agents to make sure that they avoid collisions and have other agents and obstacles stay in their sensing ranges.

Agent i can avoid collisions with other agents and obstacles if:

$$\|x_i - x_j\| \geqslant D_c \quad \forall j \in V \setminus \{i\} \tag{3.2}$$

$$\|x_i - x_o\| \geqslant D_o \quad \forall x_o \in O \tag{3.3}$$

where D_c and D_o are constant values based on real-life parameters assuming the agents have non-zero dimensions.

Agent i can sense other agents if:

$$\|x_j - x_i\| \leqslant D_s \tag{3.4}$$

$$\|q - x_o\| \geqslant D_l \quad \forall q \in L_{ij} \tag{3.5}$$

where the line of sight between i and j and L_{ij} is defined as $L(p,q) := (1-\lambda)p + \lambda q, \quad \forall \lambda \in [0,1]$, so an agent j is sensed by agent i if it falls into its sensing range and there exists no obstacles or other agents between them.

Agent i senses an obstacle point in the environment if:

$$\|x_o - x_i\| \leqslant D_s \tag{3.6}$$

$$\|x_o - x_i\| \geqslant \|q - x_i\| \quad \forall q \in \bar{L}(x_i, x_o) \cap O \tag{3.7}$$

where the line of sight between i and the obstacle point x_o extends as a ray to make sure this obstacle point is not obstructed by another one as illustrated in Figure 3.2. These two conditions are similar to the ones needed to sense the agents.

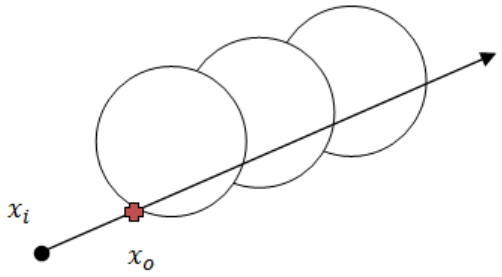

FIGURE 3.2: The highlighted obstacle point is sensed since no other obstacle point is between itself and the agent

3.1.3 Dynamics of the Network Topology

By establishing these conditions, we can define our sensing network graph $G_s := (V, E_s(x^k))$, where $E_s(x^k)$ is the set of edges of the sensing graph at time k and construct our sensing Laplacian matrix $L_s(x^k)$ where

$$s_{ij} := \begin{cases} -1 & \text{if (3.2), (3.3), (3.4), and (3.5) hold} \\ 0 & \text{Otherwise,} \end{cases} \tag{3.8}$$

for $i \neq j$ and $s_{ii} = -\sum_{j \in S_i^\sigma} s_{ij}$ where S_i, the set of agents that agent i can sense, is defined as $S_i(x^k) := \{j | (i,j) \in E_s(x^k)\}$.

If the network sensing graph G_s is connected, each follower has a path going to the leader with a maximum length of $(N-1)D_s$. However, to maintain the connectivity of our network just through guaranteeing the preservation of G_s may become unreliable since connecting edges between agents can be lost by having very minor movements or changes in the positions of the agents, therefore, we aim to preserve connectivity of $G_n(x^k) := (V, E_n(x^k))$ where $E_n(x^k) := \{(i,j) \in E_s(x^k) | \|x_i^k - x_j^k\| \leqslant D_n\}$ and $D_c \leqslant D_n \leqslant D_s$ which is a subgraph of the sensing graph

G_s and the neighbor Laplacian matrix $L_n(x^k)$ is, for $i \neq j$,

$$l_{ij} := \begin{cases} -1 & \text{if } s_{ij} = -1 \text{ and } \|x_i^k - x_j^k\|_2 \leqslant D_n \\ 0 & \text{Otherwise,} \end{cases} \tag{3.9}$$

and

$$l_{ii} = -\sum_{j \in N_i} l_{ij} \tag{3.10}$$

It is proven in [2] that in order to preserve connectivity of G_s, a constriction to have is $(D_o^2 + D_n^2)^{1/2} \leqslant D_s$.

Assumption 3.1: The neighbor network graph G_n is connected initially at $(k = 0)$ so there exists a path from any follower to the leader of the group of agents.

Hence, when initial connectivity is assumed, a lot of papers simply aim to preserve the connectivity of the initial edge set $E_n(x^0)$. Therefore, the network topology remains fixed throughout the whole process. However, this can prove impractical in the presence of obstacles in the environment where network topology needs to change for agents to appropriately navigate through a narrow space. Therefore, to be able to navigate narrow spaces in our environment and maintain connectivity of the graph at the same time, we need to deactivate the links that hinder the navigation process and keep the other links present in $E_n(x^k)$. In this case, the set of neighbors to be preserved is defined as $N_i^\sigma(x^k) := \{j \in N_i(x^k) | \sigma_{ij}^k = 1\}$ and the following subgraph is defined as $G_\sigma(x^k) := (V, E_\sigma(x^k))$ where $E_\sigma(x^k) := \{(i,j) \in E_n(x^k) | \sigma_{ij}^k = 1\}$ and σ_{ij}^k is a systematic indicator function defined as

$$\sigma_{ij}^k := \begin{cases} 0 & \text{if edge } (i,j) \text{ is deactivated} \\ 1 & \text{otherwise} \end{cases}$$

The two cases we have for link deletion are the same as in Chapter 2 where in Figure 3.3, the link between agents i and j is removed if (2.9) and (2.10) hold for the left figure and (2.11),

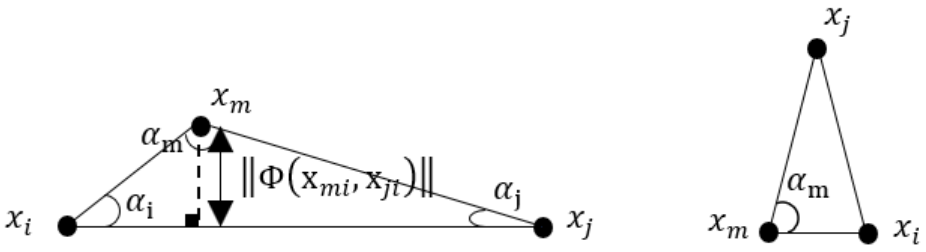

FIGURE 3.3: An illustration of the two conditions from left to right

(2.12), and (2.13) hold for the right one.

Assumption 3.2: The agents in the system have no measurement errors, therefore, if any of the two conditions for dropping links is fulfilled for one agent, it should be fulfilled for the other, i.e: if agent i drops its link with agent j at time k, then agent j drops its link with agent i at the same time step as well without any need for information exchange through communication.

In order to ensure the connectivity of our sensing network after applying the two conditions of dropping links mentioned above, we mention the theorem that [2] used to prove the connectivity of $G_\sigma(x^k)$ and the proof of this theorem is presented there as well.

Theorem 3.1 [2]: Suppose that the following assumptions hold:

1. $G_n(x^k)$ is connected,
2. All agents satisfy (3.2) and (3.3) at time k,
3. The constraint in (2.8) is satisfied,
4. D_n is chosen such that $sin(k\pi) \neq D_c/2D_n \quad \forall k = 0, 1, 2, ...,$

then $G_\sigma(x^k)$ obtained by the rule of σ_{ij}^k is connected. This means that σ_{ij} is determined such that if no edge is lost in G_σ, the connectivity of G_n is preserved.

As in Chapter 2, the leader is not bound by the same behaviors as the followers are but the leader moves with a constant speed towards a certain target. In an environment with obstacles, the leader, in this problem, moves through a path made of obstacles towards the target without considering the preservation of connectivity.

Finally, this problem can be summarized as follows: A group of autonomous agents are to move in an environment with the presence of obstacles that form a path leading towards a target only known to the leader of the group. The followers have to maintain connectivity with their neighbors and the leader while taking collision avoidance and obstacle avoidance into consideration in a leader-follower topology where links can be added or removed under certain conditions and without any communication links present between the agents.

3.2 Control Algorithm

Having explored the foundations of our system and its modeling, in this section, we explain how the agents should interact with the environment in order to achieve the desired behaviors explained previously. Since the dynamics of the agents are discrete, an agent moves through space in discrete inputs with fixed time steps, therefore our objective is to obtain the input for each time step that preserves connectivity and avoids collisions.

3.2.1 Modeling of the Potential Field

As we have established in Chapter 2, the cost function is a potential field comprised of the sum of the potential functions that lead us to the behavioral objectives where each potential function focuses on one aspect of the behaviors. The cost function here is expanded to include the obstacle avoidance behavior needed in environments with risk of collision with obstacles.

In our algorithm, the agents are subjected to the following potential field

$$U(x_i) := \sum_{j \in N_i^\sigma} c_1 f_1 + c_2 f_2 + c_3 f_3 + c_4 f_4 \tag{3.11}$$

which is specifically defined to achieve connectivity maintenance, collision avoidance, and obstacle avoidance represented by its four functions. f_1, f_2, f_3, and f_4 that constitute the potential and will be explained in this section individually and c_1, c_2, c_3, and c_4 are scalar values.

The first component f_1 takes into account the desired value of the relative distance to neighbors $j \in N_i^\sigma$ as agent i should maintain a distance value with its neighbors that maintains the connectivity and prevents collisions with them at the same time. Therefore, the functions is defined as:

$$f_1(\|x_{ij}\|_2) := w_{ij}(\|x_{ij}\|_2 - D_r)^4 \quad \|x_{ij}\|_2 := \|x_i - x_j\|_2 \tag{3.12}$$

As in Chapter 2, to preserve connectivity, a follower keeps increasing the weight of the neighbors that keep moving away from it relative to the other neighbors that do not.

We model the function f_1 as a polynomial function of the fourth degree instead of a quadratic

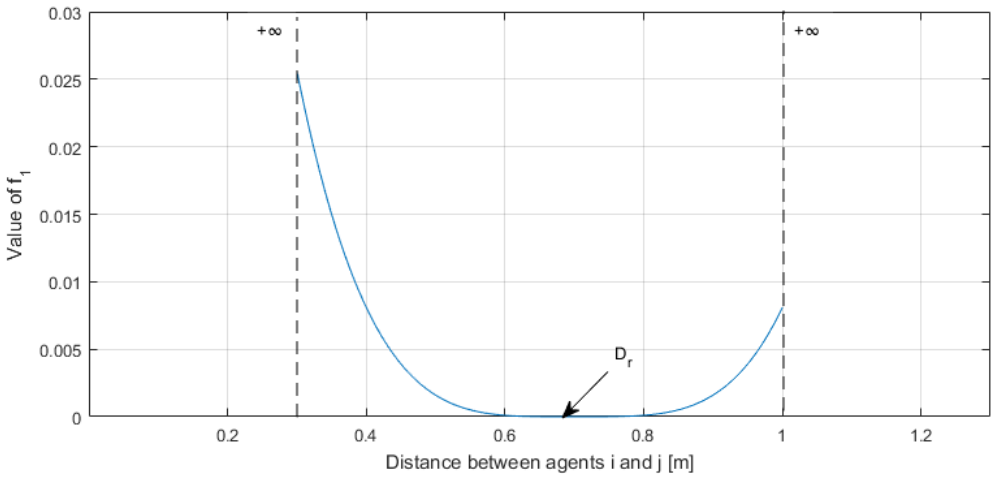

FIGURE 3.4: Plot of the potential function f_1 which shows its domain and fundamental parameters

function, as shown in Figure 3.4 in order to decrease the values of the cost imposed on the agents if the distances between themselves and their neighbors fall within an acceptable range around D_r. We can see that function f_1 takes effect only between the collision distance D_c and the neighborhood distance D_n and other than that the value of the function goes to infinity as either collision occurs or neighbors are lost for the agent.

As we have mentioned previously, since we want the importance of the functions to be toned only using the gain values c_1, c_2, c_3, and c_4, adding weights to the function f_1 has to accommodate this. Therefore, the sum of weights for agent i is defined as:

$$w_{ii} = \sum_{j \in N_i^\sigma} w_{ij} := |N_i^\sigma| \tag{3.13}$$

where $|N_i^\sigma|$ is the number of neighbors of agent i. This condition will approximately keep the total weight of function f_1 the same relative to the three other functions.

The second component f_2 is introduced for obstacle avoidance as to move away from the closest obstacle point detected at time k. The function takes into account the distance between agent i and the closest obstacle point it senses within its vicinity. In other words, the objective of this functions is to keep a minimum distance of D_{or} between agent i and the closest obstacle point denoted as o_i at time step k as illustrated in Figure **??**.

In order to calculate the closest obstacle point to agent i, we utilize geometric laws where modeling the obstacles as circles comes into play. The closest obstacle point, represented by a point on the outline of a circle, can be obtained by calculating the shortest distance between the agent represented by a point mass and the center of all the obstacles, represented by circles, sensed by agent i. Therefore, the function is defined as:

$$f_2(\|x_{io}\|_2) := (\|x_{io}\|_2 - D_{or})^2, \quad \|x_{io}\|_2 := \|x_i - o_i\|_2 \tag{3.14}$$

$$o_i := \min_{o \in O} \begin{bmatrix} c_o(1) + r\frac{x_i(1)-c_o(1)}{\|x_i - c_o\|_2} \\ c_o(2) + r\frac{x_i(2)-c_o(2)}{\|x_i - c_o\|_2} \end{bmatrix} \tag{3.15}$$

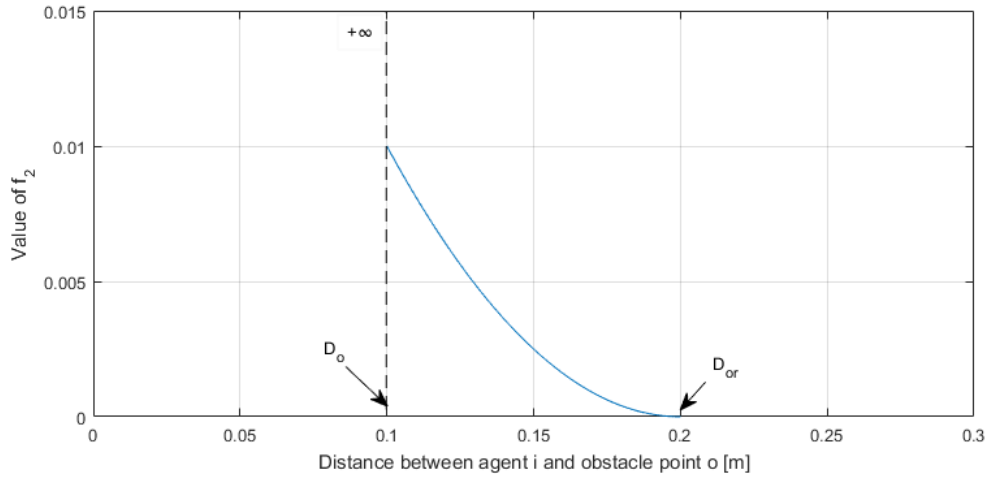

FIGURE 3.5: Plot of the potential function f_2 which shows its domain and fundamental parameters

We model function f_2 as a quadratic function whose maximum value occurs at the collision distance D_o between agent i and the obstacle point and this value decreases till it reaches its minimum when the agent lies at an acceptable distance D_{or} from the obstacle point then the value continues as minimum as the distance increases. The domain of distance that function f_2 deals with is higher than the collision distance D_o and any values lower than that makes the function go to infinity as it is shown in Figure 3.5.

After explaining the functions f_1 and f_2, one can notice the difference in the power of the function, where in function f_1, the function is raised to the fourth power, while for function f_2, the functions is a simple quadratic function.

Since we introduce obstacles in our environment in this chapter, we do not want function f_1 to be an inconvenience to the other desired behaviors such as obstacle avoidance. Therefore, in our modeling, we give a margin of tolerance to the inter-agent desired distances which we choose to be 60% further from D_r in both directions of the function.

Due to the nature of the polynomial functions raised to an even power, the values on the x-axis flatten around the minimizer to have corresponding low values on the y-axis as the power of

the functions increase and then grow steeper as we move further from the minimizer as shown in Figure 3.6(a).

In order to determine the value of the even power n to be used in function f_1, we check the relation between the power of the function f_1 and the area till which the function f_1 has a lower cost than function f_2. In other words, we want $f_1 \leqslant f_2$ where:

$$\|x_{ij}\|_2 \leqslant D_r - p(D_r - D_c) \tag{3.16}$$

$$\|x_{io}\|_2 \leqslant D_{or} - p(D_{or} - D_o) \tag{3.17}$$

where $0 \leqslant p \leqslant 1$. By substituting from (3.12) and (3.14), we have:

$$(D_r - p(D_r - D_c) - D_r)^n \leqslant (D_{or} - p(D_{or} - D_o) - D_{or})^2 \tag{3.18}$$

$$(p(D_c - D_r))^n \leqslant (p(D_o - D_{or}))^2 \tag{3.19}$$

$$p^{n-2} \leqslant \frac{(D_o - D_{or})^2}{(D_c - D_r)^n} \tag{3.20}$$

At $D_c = 0.3$, $D_r = 0.7$, $D_o = 0.1$, and $D_{or} = 0.2$, we can see that at $n = 2$, the condition becomes invalid and the only n that keeps the percentage p within the range where $p \leqslant 0.625$ is $n = 4$. Therefore, we choose the function f_1 to be raised to the fourth power because:

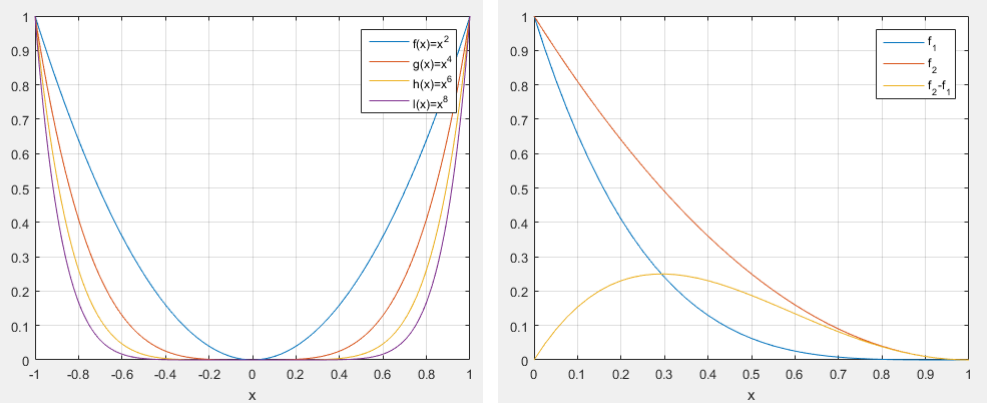

FIGURE 3.6: (a) A plot of different polynomial functions and (b) the difference in cost between the two functions

1. it has lower cost than the quadratic function f_2 in the tolerance margin defined above.

2. it has a smaller margin of tolerance than the other functions with higher powers.

In order to show the cost difference between functions f_1 and f_2, we scale the x-axis values and y-axis values to have the same domain and range as shown in Figure 3.6(b). We can see that the cost difference keeps increasing till it reaches 62.5% far from the minimizer in both functions then it starts decreasing again since the agent needs to perform the behaviors implied by both functions simultaneously.

N.B.: Our choice for function f_1 instead of a standard quadratic function is based solely on the specific model used in this work, and is not a desirable choice in general.

The third component f_3 is included in order to avoid obstruction of the line of sight (LOS) between the agent and its neighbors. The function f_3 takes into consideration the lines of sight between agent i and its neighbors and obtains the line of sight which is closest to an obstacle point at time k. Henceforth, the objective of function f_3 is to keep this shortest line of sight from being obstructed by obstacles in order to maintain the connectivity between agent i and that agent.

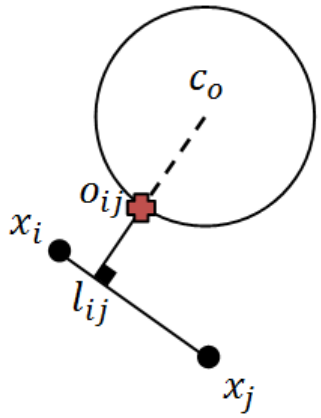

FIGURE 3.7: An illustration of the closest obstacle point to an LOS

Based on geometric laws, the shortest distance between a line and a circle is a perpendicular to that line. In other words, the shortest distance between the line of sight between agent i and a neighbor, represented by a line segment, and the closest obstacle point is obtained by calculating the shortest perpendicular line segment extending from the point on the LOS denoted by l_{ij} and the obstacle point denoted by o_{ij} as illustrated in Figure 3.7 among all neighbors of agent i and obstacles sensed by the agent. Therefore, function f_3 is defined as:

$$f_3(\|x_{ij^*o}\|_2) := (\|x_{ij^*o}\|_2 - D_{lr})^2, \quad \|x_{ij^*o}\|_2 := \min_{j \in N_i^\sigma} \|l_{ij} - o_{ij}\|_2 \tag{3.21}$$

$$l_{ij} := \begin{bmatrix} \frac{c_o(1)+m_{ij}(c_o(2)-b_{ij})}{1+m_{ij}^2} \\ m_{ij} l_{ij}(1) + b_{ij} \end{bmatrix} \tag{3.22}$$

$$o_{ij} := \begin{bmatrix} c_o(1) \pm \frac{r}{\sqrt{1+m_{ij}^2}} \\ c_o(2) + \frac{c_o(1)-o_{ij}(1)}{m_{ij}} \end{bmatrix} \tag{3.23}$$

where

$$m_{ij} = \frac{x_j(2) - x_i(2)}{x_j(1) - x_i(1)} \tag{3.24}$$

$$b_{ij} = x_i(2) - (m_{ij} x_i(1)) \tag{3.25}$$

as the line between agent i and agent j is defined as $y = m_{ij}x + b_{ij}$

The function f_3 is modeled the same as f_2 as a quadratic function whose maximum value is at the point of losing connection between the two agents D_l. The value of the function decreases till it reaches the minimum at an acceptable distance of D_{lr} between the link and the obstacle point and continues as minimum value as the distance increases. The function f_3 becomes unbounded if its argument is lower than D_{lr} as shown in Figure 3.8.

The fourth component f_4 is introduced to make the group more cohesive as long as the behaviors desired from the agents remain intact. So, the function f_4 resembles the function f_2 in

Chapter 2 in its objective. For example, assume agent p is not a neighbor for agent i and agent i has little or no forces in terms of connectivity preservation, collision avoidance, or obstacle avoidance that are pushing it away from agent p, agent i moves closer to agent p due to the effect of function f_4 which is defined as follows:

$$f_4(\|x_{ij}\|_2) := \begin{cases} (\|x_{ij}\|_2 - D_n)^2 & \text{if} \quad D_s \geqslant \|x_{ij}\|_2 > D_n, O_i = \Phi \\ 0.1(\|x_{ij}\|_2 - D_n)^2 & \text{if} \quad D_s \geqslant \|x_{ij}\|_2 > D_n, O_i \neq \Phi \end{cases} \tag{3.26}$$

where O_i is defined as the set of obstacles that agent i senses at time step k.

We see in (3.26) that the effect of the potential function f_4 drops to tenth of its value when obstacles exist within the sensing range of agent i in order not to compromise the obstacle-avoidance capability of the system.

The function f_4 is modeled as a quadratic function whose minimum value is at D_n because, as the distance goes smaller than that, function f_4 ceases to have an effect regarding this particular agent and function f_1 becomes the main force affecting in regards to connectivity preservation. Unlike the previous three functions, function f_4 does not go to infinity if the domain goes beyond the sensing distance D_s as it is not a crucial part of the connectivity preservation behavior

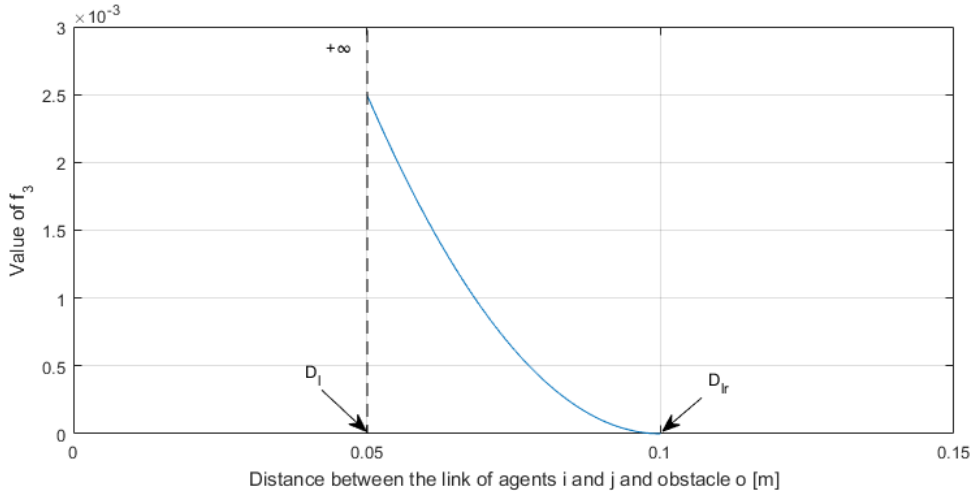

FIGURE 3.8: Plot of the potential function f_3 which shows its domain and fundamental parameters

of the agents to keep sensing agents that are not within its neighborhood set as it is shown in Figure 3.9.

Since the four functions that constitute our potential field have ranges of different values, we need to re-scale the functions so that they all have the same values for their minimums and maximums. Therefore, they all can have the same influence on the agent as they go towards their extremes.

If we desire to change the influence of one of the functions in relation to the others, one can do so by changing the values of c_1, c_2, c_3, and c_4 of $U(x_i)$ in (3.11).

3.2.2 Obtaining the Input of the Agents

Having completed the modeling of the potential field, we are ready to acquire the inputs for the agents is the next step logically.

In this subsection, we go briefly through the gradient descent algorithm used in [2] to highlight the difference between our algorithm and theirs and to make it easier to understand how both methodologies perform differently under the same conditions.

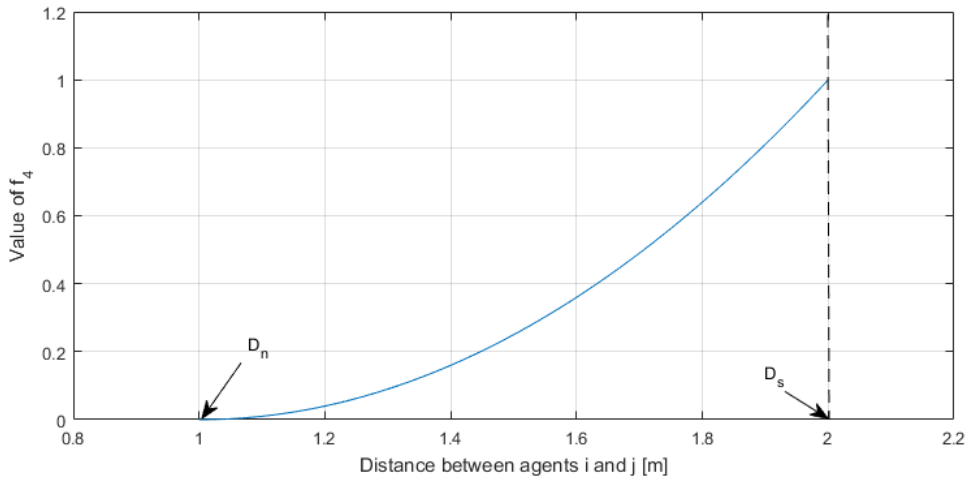

FIGURE 3.9: Plot of the potential function f_4 which shows its domain and fundamental parameters

In [2], the direction and magnitude of the input vector are obtained separately. The direction of the input vector is denoted by $v_i := -\nabla_{x_i}\Psi_i(x), \quad i \in V \setminus N$ where $\Psi_i(x)$ is the artificial potential field set by the functions modeled and chosen by [2]. In their model, as we have mentioned previously, the direction of the input vector of the leader N is already predetermined based on the position of the target and the trajectory the leader takes to reach the target.

Regarding the magnitude of the input vector denoted by $u_i(k)$, it needs to satisfy four conditions during the transition of the agent from the state $x_i(k)$ to $x_i(k+1)$:

1. The maximum distance condition for each $j \in N_i^{\sigma}(x^k)$
2. The inter-agent collision avoidance condition for each $j \in S_i(x^k)$
3. The obstacle avoidance condition for each $x_o \in O_i(x^k)$
4. The LOS preservation condition for each $j \in N_i^{\sigma}(x^k)$

These conditions are stated to ensure that the input acquired by the agent not only make the agent achieve the desired behaviors but also achieve them while maintaining convexity i.e.: the agent remains safe while going from the position $x_i(k)$ to $x_i(k+1)$. Therefore, the upper bounds obtained from the four above conditions were denoted by $\bar{u}_i^{con1}$ and $\bar{u}_i^{con2}$ for the first condition, $\bar{u}_i^{col}$, $\bar{u}_i^{obs}$, and $\bar{u}_i^{los}$ for the second, third, and fourth condition respectively.

Although gradient descent algorithm is used extensively in papers that model the environment as a potential field to navigate as in [11, 39, 41, 42, 46], we utilize the model predictive controller in our algorithm, and our problem can be described by the following formulation:

$$\text{minimize} \qquad J = U(x_i) + u_i^T R u_i \tag{3.27}$$

$$\text{subject to} \qquad x_i^{k+1} = x_i^k + u_i^k \tag{3.28}$$

Here J is the cost function and R is a positive-definite matrix as the cost of the input is taken into consideration for each time step to avoid unnecessarily large shifts in the positions of the

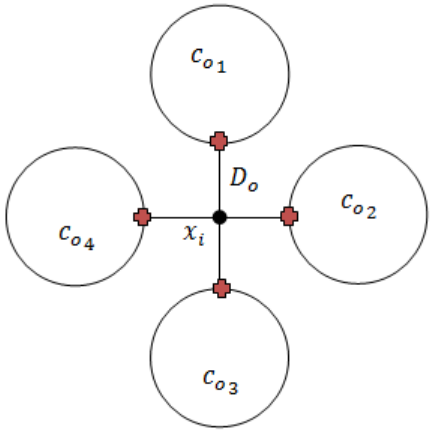

FIGURE 3.10: Agent i is surrounded by four obstacles with equal distances from it of value D_o

agents which is more in line with real-life restrictions. The big $\mathcal{O}$ notation shows that the computational complexity of this solution is of order $\mathcal{O}(N^3 + NO^2)$ where N is the number of agents and O is the number of obstacles.

In certain cases, our algorithm fails to reach a solution where the agents experience deadlock and thus they can no longer continue the mission without losing connectivity, colliding with obstacles, or colliding with each other, therefore, we need to inspect these cases. We have already inspected the effect of changing the weight values of the links between the agents in Chapter 2 in Theorem 2.2.

We can extend Theorem 2.2 by applying the same condition for function f_2 in the potential function $U(x_i)$ where

$$c_2 f_2(\|x_{io}^k\|_2) \leqslant 1 \tag{3.29}$$

$$c_2(\|x_{io}^k\|_2 - D_{or})^2 \leqslant 1 \quad \text{from (3.14)} \tag{3.30}$$

We can ensure that this condition holds all the time since the deadlock cases for this function depend solely on the obstacle structure in the environment and as shown in Figure 3.10. Here agent i is about to collide with obstacles that surround it from all directions, any action that

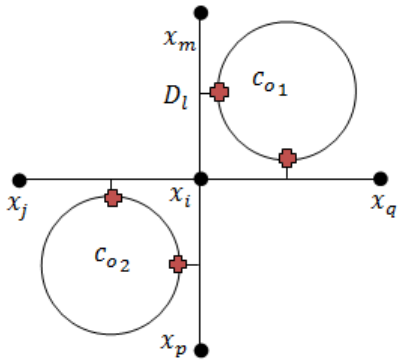

FIGURE 3.11: A deadlock case for agent i based on f_3

agent i takes, in this case, will lead to it colliding because of the parameters of the system: obstacle collision distance D_o and the obstacle radius r. Therefore, it is not possible for the agent to end up in a deadlock position for function f_2 unless it started in that position initially at ($K = 0$).

The same extension of theorem 2.2 can be done for function f_3 where

$$c_3 f_3(\|x^k_{ij^*o}\|_2) \leqslant 1 \tag{3.31}$$

$$c_3(\|x^k_{ij^*o}\|_2 - D_{lr})^2 \leqslant 1 \quad \text{from (3.21)} \tag{3.32}$$

We demonstrate an example of the cases where agent i faces deadlock based on function f_3 in Figure 3.11 where any movement leads agent i to lose connection with at least one neighbor because the lines of sight with the neighbors are at distances of D_l with the obstacles and will be obstructed by these obstacles in case of any input.

Finally, we can conclude that agent i preserves connectivity with its neighbors and avoids collision with them and the obstacle points if:

$$\sum_{j \in N_i^\sigma} c_1 f_1 + c_2 f_2 + c_3 f_3 \leqslant |N_i^\sigma| + 2 \tag{3.33}$$

by adding (2.42),(3.29), and (3.31). This means that agent i does not find an input that satisfies all the desired behaviors and deadlock occurs if and only if a combination of the three functions f_1, f_2, and f_3 violate the inequality condition set in (3.33).

3.3 Simulation Results

In order to examine the effectiveness of the proposed algorithms explained in the previous chapter, we run various simulations in Matlab^{TM} environment using the *fmincon* function. We use the default algorithm used in the *fmincon* function which is the interior-point method as it satisfies our requirements in this work regarding the computation time and the number of agents.

The parameters in our formulation can be classified into two types:

1. Physical parameters that can not be changed unless we change the physical structure of the agents or the environment
2. Artificial parameters that we choose for the models and algorithms presented in our work

The parameters that fall into the first category are:

(a) The maximum sensing range (D_s) of the agents which depends on their physical sensors.

(b) The minimum distance between the agents and each other (D_c) at which collision occurs which depends on the physical structure of the agents.

(c) The minimum distance between the agents and the obstacles (D_o) at which collision occurs which depends on the physical structure of the agents and the obstacles in the environment.

(d) The minimum distance between the line of sight between two agents and the obstacles (D_l) at which the vision between the agents is obstructed. This parameter depends on the placement of the sensors on the agents and the structure of the obstacles in the environment.

We have $D_s = 2m$, $D_c = 0.3m$, $D_o = 0.1m$, and $D_l = 0.05m$ which are the same values as in [2]. And the parameters that fall into the second category are:

(a) The number of the agents (N)

(b) The maximum distance between agents (D_n) at which agents are considered neighbors with one another

(c) The distance desired to be between the agents and each other (D_r)

(d) The minimum distance desired to be between the agents and the obstacles (D_{or}) at which function f_2 takes effect

(e) The minimum distance desired to be between the line of sight between two agents and the obstacles (D_{lr}) at which function f_3 takes effect

(f) The tolerance values ε_n and ε_c that are taken into account when measuring the distances between the agents in the second condition of link deactivation

(g) The gain values c_1, c_2, c_3, and c_4 of the functions in the potential field

We choose $N = 5$, $D_n = 1m$, $D_r = 0.7m$, $D_{or} = 0.2m$, $D_{lr} = 0.1m$, $\varepsilon_n = \varepsilon_c = 0.1m$, $c_1 = 9.77$, $c_2 = 100$, $c_3 = 400$, $c_4 = 0.25$, and $R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$ so that the functions have a balanced effect on the inputs of the agents.

In our simulations, we run the algorithm on different scenarios changing both the conditions of the environment and the agents which are:

1. The angles of the path that the agents go through towards the target (ϕ_i) where i is the number of the turns that the path takes.

2. The length of the first straight line of the path before any turns (l_o) which is the most crucial in the path.

3. The width of the path (2η) that the agents pass through where η is the distance between the leader agent and the obstacles on one side of the path.

4. The speed of the leader or the input that the leader has each time step (u_N) which has a constant magnitude for the entire run.

We show in Figure 3.12 how the agents navigate through the environment based on our algorithm for $\phi_i = 0^\circ$ (i.e.: no path turns), $l_o = N$, $2\eta = 0.5m$, and $u_N = 0.1m$. We see how links between the agents are deactivated as they go through the narrow path. However, due to the constant speed of the leader N, cohesion by adding links is not achieved unless the leader turns instead of keeping to move in a straight line.

In Figures 3.13 and 3.14, the path has turns of $\phi_i = 30^\circ$ and $\phi_i = 90^\circ$ respectively. In both simulations, the length of the path before the first turn is $l_o = 1$ which means that the leader takes a turn before most of the followers even go inside the path, therefore, it is up to the follower right behind the leader to keep the connection of the network. It should be pointed out that the weights assigned to the leader and the rest of the followers play a huge part in that since the leader is not concerned with connectivity preservation, so our network should act like a string that is being pulled towards the direction of the leader. In Figure 3.13, the width of the path is $2\eta = 0.45m$ and the speed of the leader is $u_N = 0.05m$ per time step. In Figure 3.14, the width of the path is $2\eta = 0.4m$ and the speed of the leader is $u_N = 0.025m$ per time step.

When we compare our algorithm with the algorithm presented in [2], we see that the percentages of cases without deadlock and their distributions differ as shown in Figure 3.15. For narrower and direct paths, we note that the algorithm in [2] suffers from less deadlock cases than our algorithm because of the input constraints put on the leader, but as the path gets wider, it is

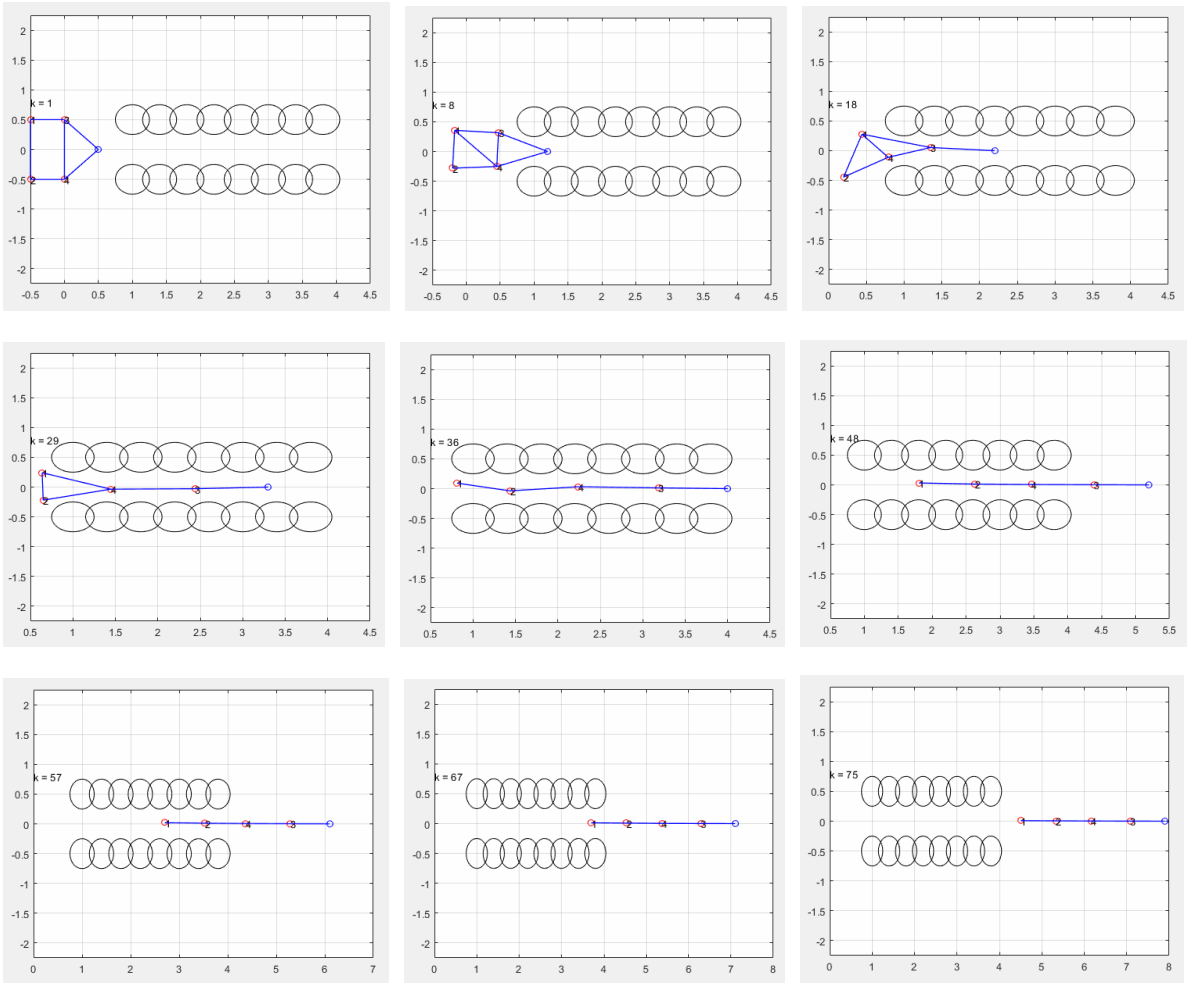

FIGURE 3.12: Snapshots for our algorithm with no turns $\phi = 0°$, the gap size of the path is equal to $0.5m$, and the speed of the leader is $u_N = 0.1m$ per time step

shown that both algorithms perform equally well. One important thing to note is, for sharper turns, our algorithm begins performs better for relatively narrower paths. This shows that even when connectivity constraints are put on the leader, some environment configurations can still cause deadlock for the network of agents. An obvious drawback of our algorithm is that the faster the leader goes towards the target, the more deadlock cases the agents will experience until it reaches a points where the agents will face deadlock cases for all the scenarios explored.

The main aspect that is enhanced in our work is that the network of agents as a whole takes fewer time steps to reach the target depending on the structure of the environment. This is

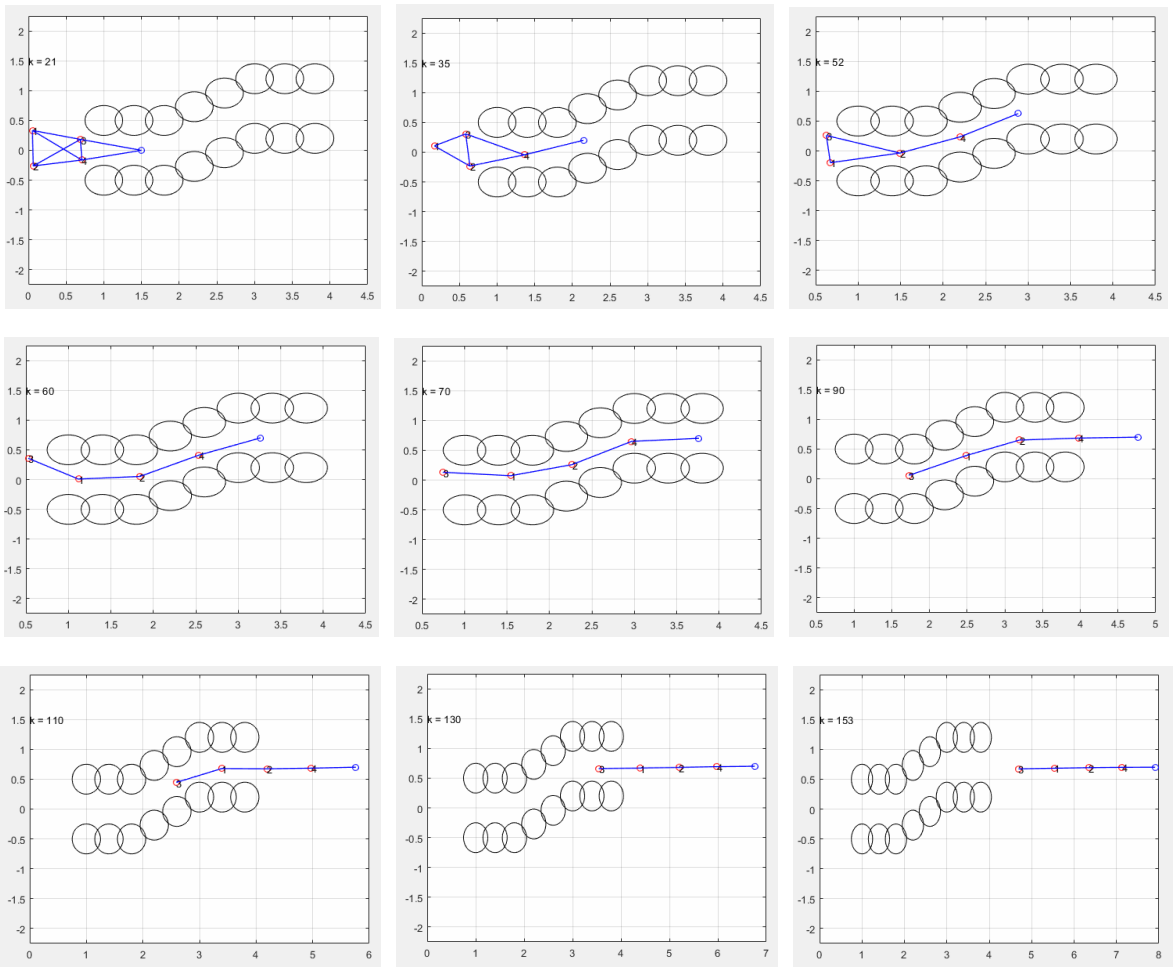

FIGURE 3.13: Snapshots for our algorithm with two turns of angle $\phi = 30°$, the gap size of the path is equal to $0.45m$, and the speed of the leader is $u_N = 0.05m$ per time step

shown in Figure 3.16 where the agents move much faster than the agents utilizing the algorithm proposed in [2] despite our algorithm facing more deadlock cases in narrower paths. We note that, in the algorithm in [2], the time steps taken to reach the target decreases as the path gets wider until it reaches a certain width where the number of time steps increases slightly then starts decreasing again. This small peak is attributed to going back and forth with link addition and removal as the agents enter the path which is aided by the small increments the leader moves with in order to preserve connectivity.

This advantage that our proposed algorithm has over the algorithm in [2] can be attributed partially to the connectivity requirement that is placed on the leader in [2] which is explained

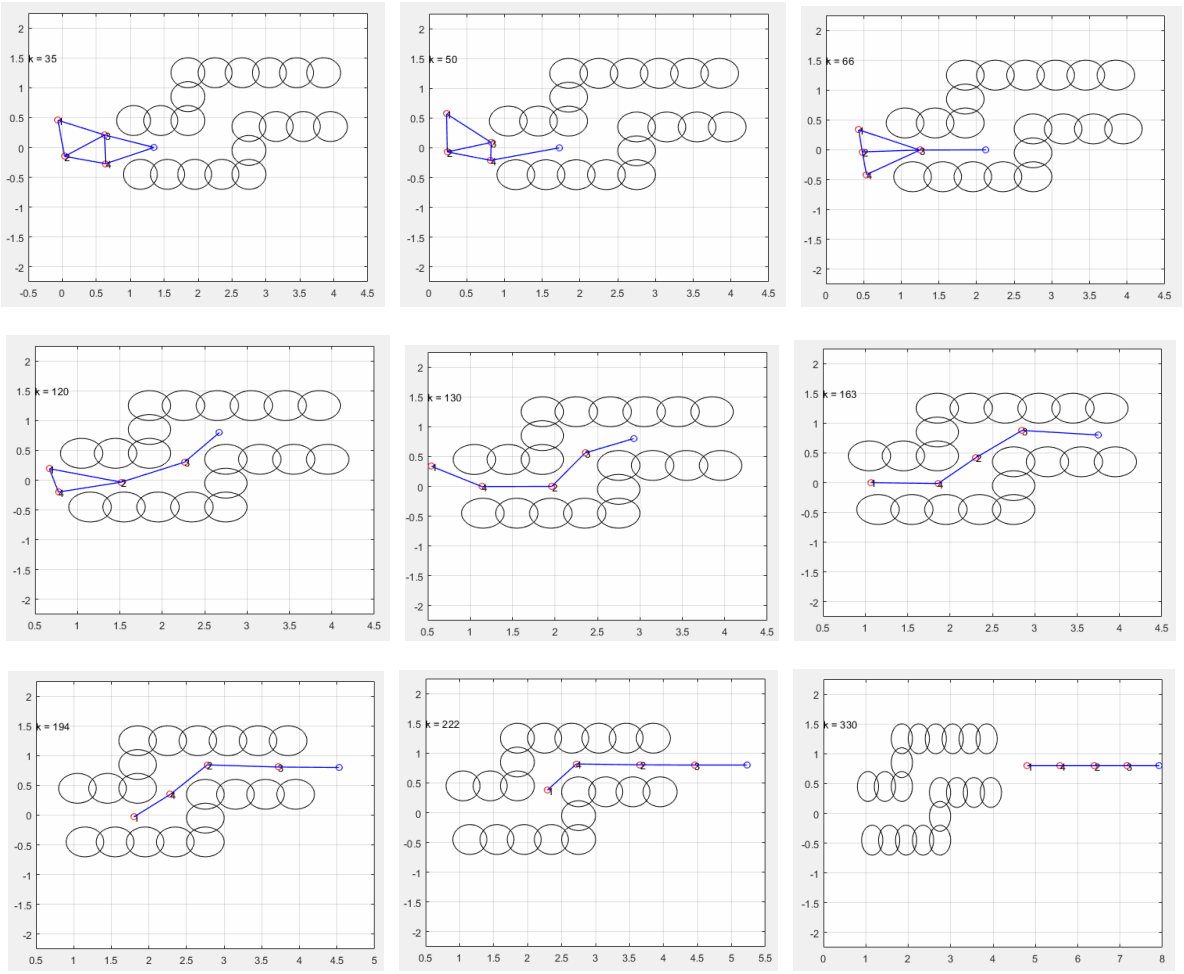

FIGURE 3.14: Snapshots for our algorithm with two turns of angle $\phi = 90°$, the gap size of the path is equal to $0.4m$, and the speed of the leader is $u_N = 0.025m$ per time step

in the two theorems below:

Theorem 2 [2]: Suppose that collision avoidance constraints in (2.2) and (2.3) are satisfied for all agents at time k. Then $u_i(k)$ defined by

$$u_i(k) := \begin{cases} \bar{u}_i(k)\frac{v_i(k)}{\|v_i(k)\|_2}, & \text{if } \|v_i(k)\|_2 > \bar{u}_i(k) \\ \|v_i(k)\|_2 & \text{Otherwise,} \end{cases} \tag{3.34}$$

satisfies the four conditions for the input upper bounds mentioned in the previous chapter for each agent $i \in V$, if

$$u_{max} \leqslant D_o - D_l, \quad (D_o^2 + D_n^2)^{1/2} \leqslant D_s \tag{3.35}$$

in addition to (2.10).

Theorem 3 [2]: In addition to the assumptions in Theorem 2, we assume

$$u_{max} \leqslant min\left\{\frac{D_s}{2}, (D_o^2 - D_l^2)^{1/2}\right\} - \frac{D_c}{2} \tag{3.36}$$

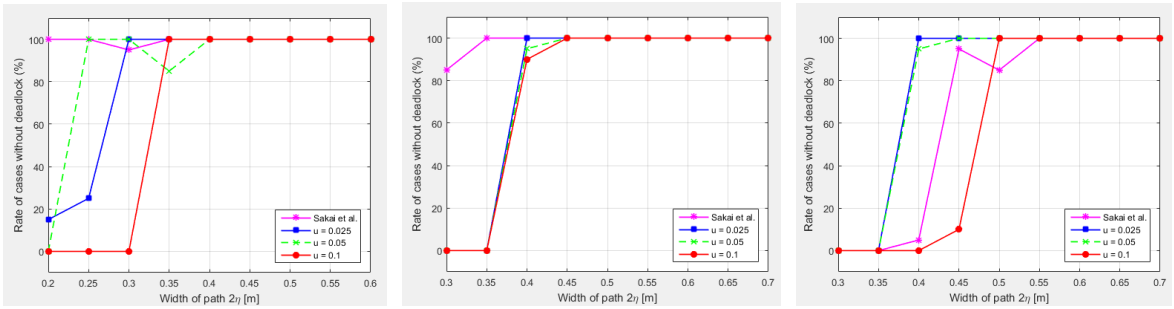

FIGURE 3.15: A comparison of the path width and the deadlock percentage between the algorithm from [2] and three different speeds of the leader in ours for $\phi_i = 0°$, $\phi = 30°$, and $\phi = 90°$ from left to right

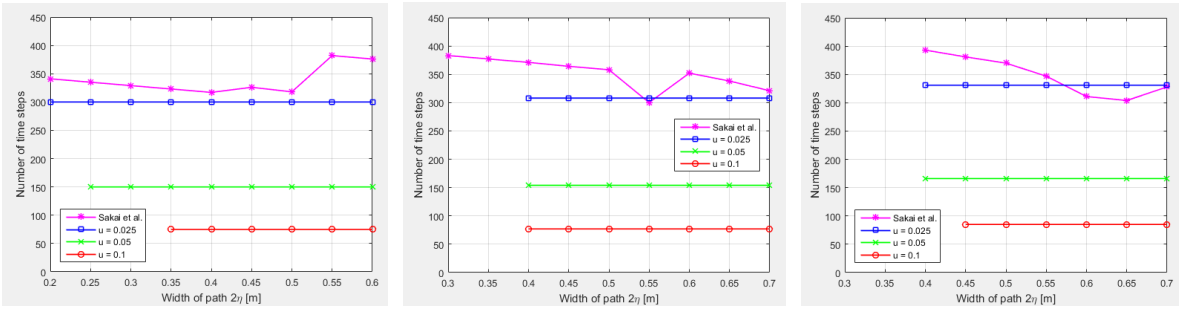

FIGURE 3.16: A comparison of the path width and time steps taken to reach the target between the algorithm from [2] and three different speeds of the leader in ours for $\phi_i = 0°$, $\phi = 30°$, and $\phi = 90°$ from left to right

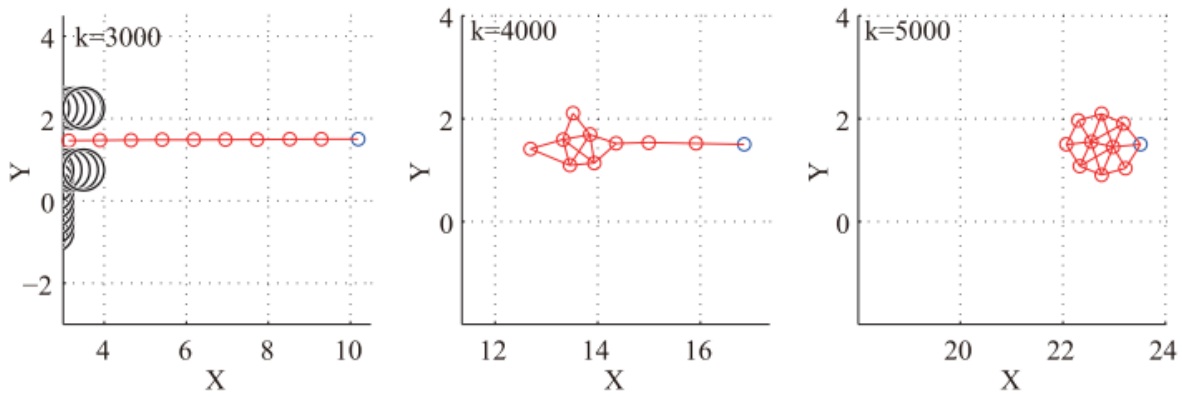

FIGURE 3.17: The agents in [2] are regrouping after the last follower exits the path made of obstacles

Then, the interagent collision avoidance condition i.e.: the second condition for the input upper bound is guaranteed for each $j \in V \setminus S_i(k)$. Furthermore, if

$$u_{max} \leqslant D_s - D_o, \tag{3.37}$$

the obstacle avoidance condition i.e.: the third condition for the input upper bound is guaranteed for each $x_o \in O \setminus O_i(k)$. Therefore, the $\bar{u}_i(k)$ that is mentioned in (3.34) can be put as

$$u_{max} = min\{\bar{u}_i^{con1}, \bar{u}_i^{con2}, \bar{u}_i^{col}, \bar{u}_i^{obs}, \bar{u}_i^{los}, u_{max}\}$$

where u_{max} is based on the two above theorems and this constraint or connectivity requirement is applied on all the followers as well as the leader in the algorithm of [2].

Although our algorithm leads to the agents reaching the target faster, it also makes our agents fall short in a different aspect. In [2], due to the connectivity requirement placed on the leader, the agents start to regroup after exiting the narrow path, despite the long time it takes to start happening as shown in Figure 3.17, while, in our algorithm, the agents do not regroup unless the leader goes through certain trajectories that allow the followers to become more cohesive. This is seen specially when the leader keeps moving in a straight line after exiting the path of obstacles and does not take any turns till it reaches the target.

We use the the second smallest eigenvalue, known as the Fiedler value, of the Laplacian matrix L_n defined previously as a measure of connectivity and we show, in Figure 3.18(a), an example of the connectivity status of the agents while going through a straight path of width $2\eta = 0.2m$ with leader input equal to $u_N = 0.025$.

Finally, we need to address an important aspect in our problem which is the input cost and determine if the fulfillment of the behaviors required for the followers brings up a cost that is too high or unpractical. In Figure 3.18(b), we show the input magnitude of an agent during a run where u_N is 0.05m/step and we can see that, although there are a few peaks in the magnitude of the input of that agent, the average of the input speed of the agent is found to be 0.042 m/step which is less than the speed of the leader.

Table 3.1 shows the average input speeds of the agents in different conditions based on changing the angles of the turns in the path, the path width, and the speed of the leader. We were able to keep the magnitude of the input that the agents have for each time step relatively low or average because of the addition of the input cost term in our cost function $J(x_i)$ which was possible due to our choice of the model predictive controller instead of the gradient descent controller where we would have had to add an explicit hard input constraint that could have lead to discarding solutions that fulfill the desired behaviors for the agents.

In summary, although the algorithm in [2] can reach the solution and avoid deadlocks in environment states where the gap width is relatively small in more cases than our algorithm, we see that our algorithm, with the same initial conditions and environment state, takes much fewer

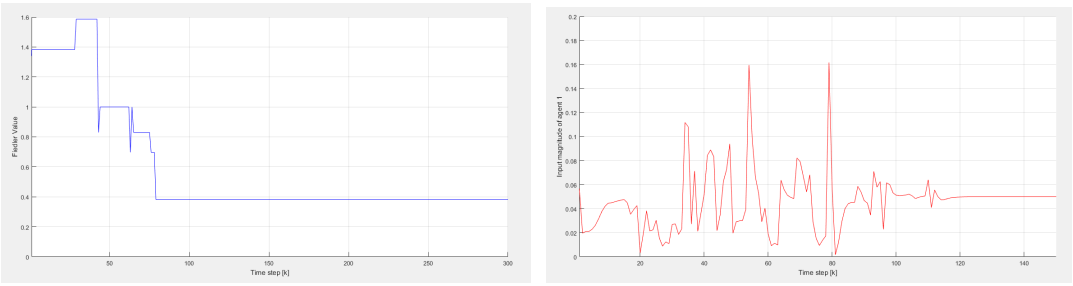

FIGURE 3.18: (a) The Fiedler value of the neighborhood connectivity matrix and (b) the input magnitude of an agent during the simulation run

Average Input Cost of the Followers				
Conditions	Agent 1	Agent 2	Agent 3	Agent 4
$\phi_i = 0$, $2\eta = 0.4$, and $u_N = 0.025$	0.0258	0.0272	0.0253	0.0253
$\phi_i = 0$, $2\eta = 0.4$, and $u_N = 0.05$	0.0468	0.0488	0.0501	0.0515
$\phi_i = 30$, $2\eta = 0.45$, and $u_N = 0.025$	0.0257	0.0278	0.0258	0.0251
$\phi_i = 30$, $2\eta = 0.45$, and $u_N = 0.05$	0.0445	0.0496	0.0478	0.0494
$\phi_i = 30$, $2\eta = 0.45$, and $u_N = 0.1$	0.0789	0.0901	0.0926	0.0991
$\phi_i = 90$, $2\eta = 0.5$, and $u_N = 0.025$	0.0279	0.0278	0.0253	0.0293
$\phi_i = 90$, $2\eta = 0.5$, and $u_N = 0.05$	0.0447	0.0469	0.0477	0.0420
$\phi_i = 90$, $2\eta = 0.5$, and $u_N = 0.1$	0.0804	0.0780	0.0859	0.0942

TABLE 3.1: Table that shows the average input magnitude of the agents in the system

time steps for the agents to reach the target through paths of moderate to large widths as a whole group with appropriate input costs.

Chapter 4

Conclusion

In this thesis, it was found that a leader-follower navigation problem of multi-agent systems (MAS) can be solved by the modeling of the behaviors of the agents and the conditions of the environment as potential functions to be implemented in a cost function employed in a model predictive controller (MPC). The MPC-based algorithm was used to navigate the agents through an obstacled environment towards an assigned target only known to the leader. There were three main desired behaviors:

1. Connectivity Preservation
2. Collision Avoidance
3. Obstacle Avoidance

These objectives are written in the form of convex functions that constitute a potential field whose value decreases as the input of the agents go towards achieving these behaviors. The topology of the agent network is dynamic and links are dropped based on two rules that serve to allow the agents to go through narrow paths in the environment without losing network connectivity. We formulate the leader to not be constrained by the behaviors imposed on the followers that is why we introduce the concept of weights given to the neighbors of agents

that affect the potential field driving their trajectories. As illustrated in the simulation results in Chapter 2 and Chapter 3, connectivity preservation is ensured using weight-based model predictive control. In Chapter 3, the simulation results show that our algorithm reaches the target faster when the agents go through paths with average or relatively large widths but faces more deadlock cases when they go through relatively narrow ones. It was shown as well that the input magnitude costs of the agents did not increase to a drawback point and they were mostly around the constant input magnitude that the leader moves with which is important in real-life application where energy conservation is taken into account.

4.1 Future Work

As it has been explained in the problem formulations, the dynamics of the agents are described as point masses with first-order discrete agent dynamics. The utilization of a model predictive controller can be made full use of by having the agents with more complex dynamics that reflect real-life agent constructions. One can as well implement our proposed algorithm in an environment with obstacles in order to compare the performance of this algorithm with the ones in the existing literature. Last but not least, the leader in this work was assumed to have constant speed throughout the simulation; however, a leader with varying speed during the operation can be inspected to see how it can affect the behavior of the followers, and to find out what adjustments are required to compensate for that.

Bibliography

[1] Wei Ren, Randal W. Beard, and Ella M. Atkins. Information consensus in multivehicle cooperative control: Collective group behavior through local interaction. *IEEE Control Systems Magazine*, pages 71–82, April 2007.

[2] Daito Sakai, Hiroaki Fukushima, and Fumitoshi Matsuno. Leader–follower navigation in obstacle environments while preserving connectivity without data transmission. *IEEE Transactions on Control Systems Technology*, pages 1233–1248, July 2018.

[3] Jeff S. Shamma. *Cooperative Control of Distributed Multi-Agent Systems*. John Wiley Sons, Ltd, Hoboken, NJ, USA, 2007.

[4] Randal Beard and Wei Ren. *Distributed Consensus in Multi-vehicle Cooperative Control: Theory and Applications*. Springer-Verlag, New York, NY, USA, 2008.

[5] Francesco Bullo, Jorge Cortes, and Sonia Martínez. *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms*. Princeton University Press, Princeton, NJ, USA, 2009.

[6] Zhihua Qu. *Cooperative Control of Dynamical Systems: Applications to Autonomous Vehicles*. Springer-Verlag, New York, NY, USA, 2009.

[7] Magnus Egerstedt and Mehran Mesbahi. *Graph Theoretic Methods in Multiagent Networks*. Princeton University Press, Princeton, NJ, USA, 2010.

[8] He Bai, John Wen, and Murat Arcak. *Cooperative Control Design: A Systematic, Passivity-Based Approach*. Springer-Verlag, New York, NY, USA, 2011.

[9] Wei Ren and Yongan Cao. *Distributed Coordination of Multi-agent Networks Emergent Problems, Models, and Issues*. Springer-Verlag, New York, NY, USA, 2011.

[10] Zhisheng Duan and Zhong-Kui Li. *Cooperative Control of Multi-Agent Systems: A Consensus Region Approach*. CRC, Boca Raton, FL, USA, 2014.

[11] Reza Olfati-Saber, J. Alex Fax, and Richard M. Murray. Consensus and cooperation in networked multi-agent systems. in *Proceedings of the IEEE*, pages 215–233, January 2007.

[12] Richard M. Murray. Recent research in cooperative control of multivehicle systems. *Journal of Dynamic Systems, Measurement, and Control*, pages 571–583, September 2007.

[13] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial Informatics*, pages 427–438, February 2013.

[14] Steffi Knorn, Zhiyong Chen, and Richard H. Middleton. Overview: collective control of multiagent systems. *IEEE Transactions on Control of Network Systems*, pages 334–347, December 2016.

[15] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, pages 25–34, July 1987.

[16] Yiguang Hong, Jiangping Hu, and Linxin Gao. Tracking control for multi-agent consensus with an active leader and variable topology. *Automatica*, pages 1177–1182, July 2006.

[17] Yiguang Hong, Guanrong Chen, and Linda Bushnell. Distributed observers design for leader-following control of multi-agent networks. *Automatica*, pages 846–850, March 2008.

[18] Jiangping Hu and Gang Feng. Distributed tracking control of leader–follower multi-agent systems under noisy measurement. *Automatica*, pages 1382–1387, August 2010.

[19] Wenjie Dong. On consensus algorithms of multiple uncertain mechanical systems with a reference trajectory. *IEEE Transactions on Automatic Control*, pages 2023–2028, September 2011.

[20] Jie Mei, Wei Ren, and Guangfu Ma. Distributed coordinated tracking with a dynamic leader for multiple euler-lagrange systems. *IEEE Transactions on Automatic Control*, pages 1415–1421, June 2011.

[21] Wei Zhu and Daizhan Cheng. Leader-following consensus of second-order agents with multiple time-varying delays. *Automatica*, pages 1994–1999, December 2010.

[22] Herbert G. Tanner, Ali Jadbabaie, and George J. Pappas. Stable flocking of mobile agents, part i: fixed topology. in *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 2010–2015, December 2003.

[23] Wei Ren and Randal W. Beard. Consensus seeking in multiagent systems under dynamically changing interaction topologies. *IEEE Transactions on Automatic Control*, pages 655–661, May 2005.

[24] Amir Ajorlou, Mohammad Mehdi Asadi, Amir G. Aghdam, and Stephane Blouin. Distributed consensus control of unicycle agents in the presence of external disturbances. *Systems Control Letters*, pages 86–90, May 2015.

[25] Ali Jadbabaie, Jie Lin, and A. Stephen Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, pages 988–1001, June 2003.

[26] Herbert G. Tanner, Ali Jadbabaie, and George J. Pappas. Stable flocking of mobile agents part ii: Dynamic topology. in *Proceedings of The 42nd IEEE Conference on Decision and Control*, pages 2016–2021, December 2003.

[27] Ji Xiang, Wei Wei, and Yanjun Li. Synchronized output regulation of networked linear systems. *IEEE Transactions on Automatic Control*, pages 1336–1341, June 2009.

[28] Youfeng Su and Jie Huang. Cooperative output regulation of linear multi-agent systems. *IEEE Transactions on Automatic Control*, pages 1062–1066, April 2012.

[29] Xiaoli Wang, Yiguang Hong, Jie Huang, and Zhong-Ping Jiang. A distributed control approach to a robust output regulation problem for multi-agent linear systems. *IEEE Transactions on Automatic Control*, pages 2891–2895, December 2010.

[30] Youfeng Su, Yiguang Hong, and Jie Huang. A general result on the robust cooperative output regulation for linear uncertain multi-agent systems. *IEEE Transactions on Automatic Control*, pages 1275–1279, May 2013.

[31] Peter Wieland and Frank Allgower. An internal model principle for synchronization. in *Proceedings of the IEEE International Conference on Control and Automation*, pages 285–290, May 2009.

[32] Zhiyong Chen. Pattern synchronization of nonlinear heterogeneous multiagent networks with jointly connected topologies. *IEEE Transactions on Control of Network Systems*, pages 349–359, December 2014.

[33] Alberto Isidori, Lorenzo Marconi, and Giacomo Casadei. Robust output synchronization of a network of heterogeneous nonlinear agents via nonlinear regulation theory. *IEEE Transactions on Automatic Control*, pages 2680–2691, October 2014.

[34] Xi Chen and Zhiyong Chen. Robust perturbed output regulation and synchronization of nonlinear heterogeneous multi-agents. *IEEE Transactions on Cybernetics*, pages 3111–3122, December 2016.

[35] Lijun Zhu, Zhiyong Chen, and Richard H. Middleton. A general framework for robust output synchronization of heterogeneous nonlinear networked systems. *IEEE Transactions on Automatic Control*, pages 2092–2107, August 2016.

[36] Reza Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on Automatic Control*, pages 401–420, March 2006.

[37] Yongcan Cao and Wei Ren. Distributed coordinated tracking with reduced interaction via a variable structure approach. *IEEE Transactions on Automatic Control*, pages 33–48, January 2012.

[38] Meng Ji and Magnus Egerstedt. Distributed coordination control of multiagent systems while preserving connectedness. *IEEE Transactions on Robotics*, pages 693–703, August 2007.

[39] Housheng Su, Xiaofan Wang, and Guanrong Chen. Rendezvous of multiple mobile agents with preserved network connectivity. *Systems and Control Letters*, pages 313–322, April 2010.

[40] Xiangpeng Li, Dong Sun, and Jie Yang. A bounded controller for multirobot navigation while maintaining network connectivity in the presence of obstacles. *Automatica*, pages 285–292, November 2012.

[41] Maria Carmela De Gennaro and Ali Jadbabaie. Decentralized control of connectivity for multi-agent systems. in *Proceedings of the 45th IEEE Conference on Decision Control*, pages 3628–3633, December 2006.

[42] Lorenzo Sabattini, Cristian Secchi, Nikhil Chopra, and Andrea Gasparri. Distributed control of multirobot systems with global connectivity maintenance. *IEEE Transactions on Robotics*, pages 1326–1332, October 2013.

[43] Michael M. Zavlanos, Herbert G. Tanner, Ali Jadbabaie, and George J. Pappas. Hybrid control for connectivity preserving flocking. *IEEE Transactions on Intelligent Transportation Systems*, pages 1255–1267, November 2009.

[44] Dongbing Gu and Zongyao Wang. Leader–follower flocking: algorithms and experiments. *IEEE Transactions on Control Systems Technology*, pages 1211–1219, September 2009.

[45] Shiyu Zhao. Affine formation maneuver control of multiagent systems. *IEEE Transactions on Automatic Control*, pages 4140–4155, December 2018.

[46] Ahmet Cezayirli and Feza Kerestecioglu. Navigation of non-communicating autonomous mobile robots with guaranteed connectivity. *Robotica*, pages 767–776, February 2013.

[47] Sebastian Ruder. An overview of gradient descent optimization algorithms. January 2016.

[48] Carlos E. Garcia, David M. Prett, and Manfred Morari. Model predictive control: Theory and practice — a survey. *Automatica*, pages 335–348, May 1989.

[49] Yadollah Rasekhipour, Amir Khajepour, Shih-Ken Chen, and Bakhtiar Litkouhi. A potential field-based model predictive path-planning controller for autonomous road vehicles. *IEEE Transactions on Intelligent Transportation Systems*, pages 1255–1267, May 2017.

[50] William B. Dunbar and Richard M. Murray. Model predictive control of coordinated multi-vehicle formations. in *Proceedings of the 41st IEEE Conference on Decision and Control*, pages 4631–4636, December 2002.

[51] Hyo-Sang Shin, Min-Jea Thak, and Hyoun-Jin Kim. Nonlinear model predictive control for multiple uavs formation using passive sensing. *International Journal of Aeronautical and Space Science*, pages 16–23, March 2011.

[52] Arthur Richards and Jonathan How. Decentralized model predictive control of cooperating uavs. in *Proceedings of the 43rd IEEE Conference on Decision and Control*, pages 4286–4291, December 2004.

[53] Ahmed T. Hafez, Anthony J. Marasco, Sidney N. Givigi, Alain Beaulieu, and Camille Alain Rabbath. Encirclement of multiple targets using model predictive control. in *Proceedings of the American Control Conference*, pages 3147–3152, June 2013.